A METHOD FOR SOLVING CERTAIN GRAPH RECOGNITION AND OPTIMIZATION PROBLEMS, WITH APPLICATIONS TO PERFECT GRAPHS

S.H. Whitesides*

TR 82-488
April 1982

Department of Computer Science
Cornell University
Ithaca, New York 14853

*On leave from Dartmouth College, Hanover, New Hampshire

This research was supported by a Dartmouth College Junior Faculty Fellowship.

A Method for Solving Certain Graph Recognition and Optimization Problems, with Applications to Perfect Graphs

S.H. Whitesides*

Department of Computer Science

Cornell University

Ithaca, N.Y. 14853

Abstract

A polynomial time membership test and solutions to the minimum coloring and maximum weight clique and stable set problems are given for certain families of graphs. In particular, it is shown that for any arbitrary hereditary family of graphs, these problems can be solved quickly for the entire family whenever they can be solved quickly for the clique cutset free members of the family. If the graphs in the family are perfect, then a similar statement holds for the minimum weight clique and stable set problems. These results are obtained by applying a polynomial time algorithm for determining whether a graph has a clique cutset. Several questions of Gavril concerning "clique separable" perfect graphs are answered, and a polynomial time recognition algorithm for Gallai's "i-triangulated" perfect graphs is given.

KEYWORDS: graph algorithms, cliques, stable sets, perfect graphs, clique separable graphs, i-triangulated graphs

---

*On leave from Dartmouth College, Hanover, N.H.

This research was supported by a Dartmouth College Junior Faculty Fellowship.

1. Introduction

In this paper, we will show how to use a polynomial time algorithm for determining whether a graph has a clique cutset (Whitesides [19]) to obtain a fast (i.e., polynomial time) test for membership in certain families of graphs. In these families, we then provide fast solutions to the following optimization problems: finding a minimum coloring, a maximum weight clique, a maximum weight stable set, a minimum weight clique cover, and a minimum weight stable set cover. In particular, we will prove "generic" theorems stating that for any given hereditary family $\mathbf{F}$ of graphs, these problems (with the exception of finding minimum weight stable set and clique covers) can be solved in polynomial time for arbitrary members of $\mathbf{F}$ whenever they can be solved in polynomial time for those members that have no clique cutsets. The same is true for recognizing members of a given family. If the graphs in $\mathbf{F}$ are perfect, then the minimum weight cover problems can also be solved by our method.

The motivation for seeking a fast way to find clique cutsets came from the problem of recognizing perfect graphs. It is well known that joining two perfect graphs along a clique contained in both is a perfection-preserving operation; hence, a natural first step in testing a graph for perfection would be to look for a clique cutset, and in case of success, to test the graphs joined together along the cutset for perfection. While we do not have a general test for perfection, we can nevertheless show how to recognize some subfamilies of perfect graphs using this idea.

As an application of our general methods, we will answer several questions raised by Gavril [10]. In particular, we will show how to solve the optimization problems mentioned above for "clique separable" graphs. These

graphs, studied by Gallai [8] and Gavril [10], are perfect and include as a proper subfamily the "i-triangulated" graphs of Gallai [8]. Gallai's graphs are defined by the property that their odd cycles of length at least five always have a pair of noncrossing chords. Chordal ("triangulated") graphs, which include bipartite graphs, are examples of Gallai graphs.

Grötschel, Lovász, and Schrijver [13] have given a polynomial time algorithm, based on the ellipsoid method, for finding a maximum weight stable set and a minimum coloring for perfect graphs in general. Hence, fast ways to solve those two problems are already known for the clique separable graphs in our application. However, the ellipsoid method approach does not illuminate the structure of the graphs.

Since there has been much work done on perfect graphs, naturally that subject is an area in which it will be interesting to apply the "generic" theorems given here. The solution to the optimization problems for clique separable graphs given in this paper shows that these theorems can be useful; no doubt there are more such examples.

It should be emphasized that the method in this paper is quite general and does not depend on having the property of perfection, except in the case that a minimum weight cover by stable sets or cliques is desired. Certainly there are other areas, such as the subject of chromatic polynomials, in which the theorems here can be applied. It is hoped that the reader will find the general approach given here to be not only conceptually simple but also structurally elucidating.

We will begin in Section 2 by reviewing the way in which any graph gives rise to a "clique cutset tree" that represents the structure of the graph with

regard to its clique cutsets. The clique cutset tree can be constructed quickly using the algorithm of Whitesides [19] for finding clique cutsets, which we briefly describe at the end of Section 2. In Section 3, we will give an algorithm for reconstructing a graph from its clique cutset tree. This is a process in which each step involves "gluing" a "primitive" subgraph onto the partially reconstructed graph at a clique. In Section 4, we will use the fact that every graph can be built up in this way to show how to solve recognition and certain optimization problems for graphs when solutions to these problems are known for their clique cutset-tree subgraphs. We will turn to applications in Section 5, where we review clique separable and Gallai graphs and show how to solve various optimization problems for these. In Section 6, we will give recognition algorithms for clique separable and Gallai graphs. Section 7 is devoted to concluding remarks.

Throughout this paper, we will assume that our graphs are finite, undirected, and connected, and that they have no loops or multiple edges. A graph with vertex set V and edge set E will be denoted G, G(V), or G(V,E). The number of vertices in G will be denoted by n. A clique of G is a set of mutually adjacent vertices of G, but need not be a maximal such set. A cutset of G is a set of vertices whose removal disconnects G, and a clique cutset is a cutset that is also a clique. The weight w(S) of the graph induced by a set S of vertices is the sum of the weights of the vertices in S.

Suppose that $\mathbf{F}$ is a hereditary family of graphs, i.e., that all subgraphs of graphs in $\mathbf{F}$ also belong to $\mathbf{F}$. The problems for which we will seek solutions are the following:

I) membership: determine whether a given graph G is a member of the family $\mathbf{F}$.

II) maximum clique: Given a graph G that belongs to **F** and whose vertices have been assigned nonnegative integer weights, find a clique in G of maximum weight.

III) minimum coloring: Given a graph G belonging to **F**, find a coloring of the vertices of G, adjacent vertices being assigned different colors, that uses the minimum possible number of colors.

IV) maximum stable set: Given a graph G that belongs to **F** and whose vertices have been assigned nonnegative integer weights, find a stable set of maximum possible weight.

In case the graphs in the family **F** are all perfect, then two additional problems that will interest us are:

V) minimum cover by cliques: Given a graph G(V) that belongs to a family **F** of perfect graphs and that has nonnegative integer weights on its vertices, find an assignment of nonnegative numbers $y_i(C_i)$ to the maximal cliques C_i such that the sum of the y_i's is as small as possible subject to the following condition. For each vertex $v \in V$,

$$\sum_{C_i \text{ s.t. } v \in C_i} y(C_i) \geq w(v) .$$

VI) minimum cover by stable sets: This problem is the same as V) above, except that "clique" is replaced by "stable set".

2. Constructing Clique Cutset Trees

Suppose that A is a cutset of G(V), and let V_i be the vertex set of any of the connected components of G(V-A). Then we say that the graph $G(V_i \cup A)$ is a child of G produced by A. (This terminology differs from that of Gavril

[10], where $G(V_i \cup A)$ is called a "leave" of G. We have changed this terminology because we will want to use the term "leaves" to mean degree 1 vertices of a tree other than the root.)

Following Gavril [10], we now construct a labeled tree T(G) from G as follows. Let the root of T(G) be labeled (G,A), where A is a minimal clique cutset of G. (See Fig. 1.) Then for each child of G produced by A, create in the tree T(G) a child of the root. Once these new vertices have been added to the tree, look for minimal clique cutsets for the corresponding subgraphs of G so that the new vertices can be labeled with a pair consisting of the name of the subgraph and the name of the clique cutset. Then continue this process, making a vertex in the tree a leaf whenever the subgraph of G to which it corresponds contains no clique cutset. Give the label of such a leaf a dash, "-", for a second component. If in using this notation it should happen that some tree vertex denoted (H,B), say, turns out to be a leaf, interpret "B" as a dash. We call this tree a clique cutset tree for G. In general, of course, a graph has several clique cutset trees.

The following observation will be needed to show that the algorithms given in later sections are fast.

Lemma 1 (Gavril [10]): Any clique cutset tree T(G) for a graph G has at most n^2 vertices, where n is the number of vertices of G.

Proof: Note that each nonleaf of T(G) has at least two children, so that for each tree vertex other than the root, there is at least one other tree vertex with the same parent. Suppose that tree vertices (H_i,A_i) and (H_j,A_j), which might be leaves, are both children of (H,A). Then there is an ordered pair of nonadjacent vertices (v_i,v_j) in G such that $v_i \in H_i - A$ and $v_j \in H_j - A$, $i \neq j$. To

each vertex (H_i,A_i) of T(G) other than the root, associate such an ordered pair (v_i,v_j). No ordered pair can possibly be associated with more than one tree vertex. Hence the number of vertices in T(G) is at most one more than twice the number of nonedges in G, so n^2 is certainly an upper bound for the size of T(G). □

Another useful fact about clique cutset trees is given in the following Lemma.

Lemma 2 (Gavril [10]): If (H',A') is a vertex in a clique cutset tree T(G) of a graph G, then A' is a clique cutset of G as well as of H'.

Proof: Note that if (H',A') is a child of (H,A) in T(G), then any clique cutset of H' is a clique cutset of H in G also. □

We now give a brief description of the algorithm of Whitesides [19] for finding whether a graph has a clique cutset in time $O(n^3)$. See [19] for further details, including a proof of correctness.

Algorithm 1 (Whitesides [19]): Finding whether a graph G(V) has a clique cutset.

Step 0) Look for a chordless cycle of length at least four. If no such cycle exists, then either n = 3 and G is a path whose middle vertex is an articulation point, or G must be a clique and has no clique cutset. If the desired cycle is found, denote its vertex set by S.

Step 1) Find a component C of G(V-S), and then find the set R consisting of each vertex in S that has at least one neighbor in C. (Since G(V) is connected, R is not empty.)

Step 2) Search for a pair of nonadjacent vertices u and v in R. If no such pair exists, R must be a clique cutset of G(V), as R separates C from the nonempty set S-R in G(V).

Step 3) Find a chordless path P whose end vertices are u and v and whose intermediate vertices all lie in C.

Step 4) Adjoin the intermediate vertices of P to S.

Step 5) Repeat steps 1) - 4) until either a clique cutset has been found or S has been enlarged to all of V, in which case G(V) has no clique cutset.

Now it is easy to give the main result of this section.

Theorem 1: A clique cutset tree can be constructed for a graph G(V) with n vertices in $O(n^5)$ time.

Proof: By Lemma 1, there are at most n^2 vertices in such a tree. By Algorithm 1, the children of each tree vertex can be found in $O(n^3)$ time. □

The time complexity given in Theorem 1 may be too pessimistic. Our main point, however, is that a clique cutset tree can be found in polynomial time.

3. Reconstructing Graphs from Primitive Subgraphs

The purpose of this section is to show how to reconstruct a graph G in $O(n^5)$ time by a sequence of "gluing" operations. Each of these operations consists of adjoining a "primitive" subgraph to a partial reconstruction R of graph G. The primitive subgraph is adjoined to R at some clique of G that is contained in both. By a primitive subgraph, we mean a connected subgraph that

has no clique cutset. We will later make use of this reconstruction process to find, given some hereditary family $\mathbf{F}$ of graphs, polynomial time solutions to problems I - IV of Section 1.

The basic idea of the reconstruction algorithm is simple. Begin with a primitive subgraph of G that is associated with a leaf of a clique cutset tree T(G) for G, and then one by one, adjoin the other primitive subgraphs associated with the leaves of T(G). The order in which this is done is such that the partially reconstructed graph R and the primitive subgraph being adjoined always have a clique in common. At each stage, R is connected. Hence the addition of a new primitive subgraph never connects several components at once.

The reconstruction algorithm will have a step in which the leaves of a clique cutset tree are numbered. These numbers will give the order in which the primitive subgraphs associated with the leaves are to be adjoined to the partial construction R of G. In this context, we will say that a maximal subtree with root r in T(G) has been <u>numbered</u> if, and only if, all its leaves have been assigned numbers.

<u>Algorithm 2</u>: Reconstruction of a graph G from primitive subgraphs.

Step 0) Check whether G is primitive. If so, stop.

Step 1) Construct a clique cutset tree T(G) for G.

Step 2) Number T(G) by the following procedure. (See Fig. 2.)

i) At each nonleaf vertex of T(G), assign some ordering to the children of that vertex. For the children of the root of T(G), there are no constraints on this ordering. For the children of

any other nonleaf vertex (H',A'), the only requirement on the ordering is that the first child be determined by the following rule. The first child (H",A") of (H',A') is the child such that in G, the subgraph H" contains the clique A associated with the grandparent (H,A) of (H",A") in T(G). (Note that H" is uniquely determined.)

ii) Set a variable r equal to that leaf of T(G) having the property that it and all its ancestors except for the root of T(G) are the first children of their parents. Also, set a counter variable i equal to 1.

iii) Number the subtree rooted at r as follows. If r is a leaf, assign it the number i, and increment i. If r is not a leaf, proceed recursively by numbering the subtrees rooted at the children of r one by one; process these subtrees in the order assigned to their roots during step i) above.

iv) If r is not the last child of its parent, set r equal to the next child of its parent, and repeat step iii). If r is the last child of the root of T(G), proceed to step 3). If r is the last child of any other vertex of T(G), note that the numbering of the maximal subtree whose root is the parent of r has just been completed. In this case, set r equal to the next child of its grandparent.

Step 3) Find the leaf (H,-) numbered 1 during step 2), and set the initial partial reconstruction R of G equal to H. Then process the remaining leaves of T(G) from the smallest numbered leaf to the largest. To

process a leaf (J",-) that is not the first child of its parent (J',A'), adjoin J" to R at the clique A', which belongs to both of them. To process a leaf (J",-) that is the first child of its parent (J',A') and that has grandparent (J,A), adjoin J" to R at the clique A, which belongs to both J" and R. (See Fig. 3.)

4. Solving Optimization Problems in Graphs

In this section, we show how the optimization problems II - VI listed at the end of Section 1 can be solved in polynomial time once these problems can be solved in polynomial time for the primitive (i.e., clique cutset-free) members of a hereditary family **F** of graphs. We begin with problems II - IV, for which the property of perfection is not needed.

Theorem 2: Let **F** be a hereditary family of graphs, and let **F'** denote the set of its primitive members. Suppose there is a polynomial time algorithm for finding a maximum weight clique for graphs in **F'**. Then there is a polynomial time algorithm for finding a maximum weight clique for graphs in **F**.

Proof: Let G be a graph in **F**, and suppose that G has n vertices. Then according to Theorem 1, a clique cutset tree T(G) can be constructed using Algorithm 1 in $O(n^5)$ time. By Lemma 1, there are at most n^2 leaves in T(G), so maximum weight cliques for each of the subgraphs of G associated with a leaf of T(G) can be found in time $O(n^2 p(n))$, where p(n) is a polynomial bound for the time to find a maximum weight clique for an n-vertex primitive graph in **F'**.

Suppose that (H,A) is a nonleaf vertex of T(G), and denote its children, some of which may be leaves, by (H_1,A_1), ..., (H_k,A_k). For $1 \le i \le k$, let C_i be

a maximum weight clique for H_i. Then the C_i of largest weight among these cliques is a maximum weight clique for H. Consequently, finding a maximum weight clique for G is simply a matter of processing T(G) from its lowest level up to its root, selecting at each nonleaf vertex the maximum weight clique found among the graphs associated with its children. This can be done in $O(n^2)$ time, so a maximum weight clique for G can be found in polynomial time. □

The argument just given in the proof of Theorem 2 applies to finding a minimum coloring, so we have the following result.

Theorem 3: Suppose there is a polynomial time algorithm for finding a minimum coloring for graphs in the set $\mathbf{F}'$ of primitive members of some hereditary family $\mathbf{F}$ of graphs. Then there is a polynomial time algorithm for finding a minimum coloring for all graphs in $\mathbf{F}$.

To find a maximum weight stable set, we use a result of Boulala and Uhry which we describe in the next Remark.

Remark 1 (Boulala and Uhry [2]): Let G(V,E) be a graph whose vertices are weighted with nonnegative integers, and suppose that G is composed of two subgraphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$, where $V_1 \cup V_2 = V$, $E_1 \cup E_2 = E$, and $V_1 \cap V_2$ is the vertex set of a clique A of G. Consider the stable sets of G_1 that contain no vertex of A, and let $\bar{S}_1$ be one of maximum weight. Also, consider for each vertex a in A the stable sets of G_1 that contain a, and let S_a be one of maximum weight. Now change the weights on the vertices in A: Assign to each vertex a the new weight $\max\{0, w(S_a) - w(\bar{S}_1)\}$. Let S_2^* be a maximum weight stable set for G_2 once the weights in A have been changed. Finally, denote by S the following set of vertices of G: If $S_2^* \cap A$ is empty, let $S = S_2^* \cup \bar{S}_1$; oth-

erwise, let $S = S_2^* \cup S_a$, where $S_2^* \cap A = \underline{a}$. According to Remark 4.3 of Boulala and Uhry [2], S is a maximum weight stable set of G with respect to the original weighting of the vertices, and its weight is $w(S_2^*) + w(\bar{S}_1)$.

Now we are ready to find maximum weight stable sets using the clique cutset tree method.

Theorem 4: Suppose there is a polynomial time algorithm for finding a maximum weight stable for graphs in the set $\mathbf{F}'$ of primitive members of some hereditary family $\mathbf{F}$ of graphs. Then there is a polynomial time algorithm for finding a maximum weight stable set for all graphs in $\mathbf{F}$.

Proof: Let G be a graph in $\mathbf{F}$. Using Algorithm 2, reconstruct G from primitive subgraphs, gluing one primitive subgraph at a time to the most recent partial reconstruction of G. At each step after the initial one, let the graph G_1 of Remark 1 be the primitive subgraph just glued onto the partial reconstruction of G, and let G_2 be the graph onto which G_1 was glued. Thus the graph G_1, in which several maximum weight stable set problems must be solved in order to compute new weights for the vertices in the clique, is primitive. Hence, $\bar{S}_1$ and the S_a's can be found in $O(p(n)n)$ time, where $p(n)$ is a polynomial bound on the time needed to find a maximum weight stable set for a primitive graph with n vertices. Then from these stable sets, compute new weights for the vertices of $A = G_1 \cap G_2$, and recursively solve the problem of finding the stable set S_2^* of Remark 1. Since there are fewer than n^2 primitive graphs involved in the reconstruction process, a maximum stable set for G can be found in polynomial time. □

Having solved optimization problems II - IV for families of graphs whose primitive members have solutions to these problems, we turn to families of

graphs whose members are all perfect. A graph G is perfect if, and only if, each induced subgraph H of G has chromatic number equal to the size of the largest clique in H. See Berge [1] for a discussion of these graphs.

Chvátal [6] has shown that precisely for the perfect graphs can the problem of finding a maximum weight stable set be formulated as the following linear programming problem:

$$\text{maximize} \quad \sum_{i=1}^{n} w_i x_i \quad \text{subject to}$$

$$Kx \leq 1$$

$$x \geq 0, \quad \text{where K is the maximal clique versus vertex incidence matrix, and the } w_i\text{'s are the weights for the vertices.}$$

If the weights w_i are nonnegative integers, solving the dual LP problem

$$\text{minimize} \quad \Sigma\, y_i \quad \text{subject to}$$

$$yK \geq w$$

$$y \geq 0$$

amounts to finding a minimum weight clique cover.

Since the complement of a perfect graph is also perfect (Lovász [15]), the problem of finding a maximum weight clique can also be formulated as an LP problem. Solving the dual of this problem amounts to finding a minimum weight cover by stable sets.

These facts, in conjunction with Theorems 2 and 4, give us a way to solve problems V and VI.

Corollary to Theorem 4: Suppose the graphs in the hereditary family $\mathbf{F}$ of Theorem 4 are all perfect. Then there is a polynomial time algorithm for finding a minimum weight clique cover for graphs in $\mathbf{F}$.

Proof: Let G be a graph in $\mathbf{F}$, and reconstruct G from primitive subgraphs, using Algorithm 2 as before. At each stage of the reconstruction, let G_1 be the last primitive piece added, and let G_2 be the partial reconstruction to which G_1 was adjoined. Let A denote the clique $G_1 \cap G_2$, and change the weights on the vertices of A as follows. To each $a \in A$, assign a new weight $\max\{0, w(S_a) - w(\bar{S})\}$, where, of all the stable sets of G_1 containing a, S_a is one of maximum weight and of all the stable sets of G_1 containing no vertex of A, $\bar{S}$ is one of maximum weight. Then, recursively find a minimum weight clique cover for G_2 once these new weights have been assigned. By LP duality, the weight of this cover will be the weight of a maximum weight stable set for G_2.

Next, change the weights on the vertices of A again. This time, assign to each $a \in A$ the nonnegative weight $w(a) - (w(S_a) - w(\bar{S}))$, where the weights are the original ones. This makes all maximum weight stable sets of G_1 containing a have weight equal to $w(\bar{S})$. Now find a minimum weight clique cover for G_1 with these new vertex weights. The weight of this cover will be equal to $w(\bar{S})$. Finally, take the union of the clique covers for G_1 and G_2. This union is indeed a clique cover for G. The cover has minimum weight because its weight is equal to that of a maximum weight stable set for G. Again, the subgraph in which multiple subproblems must be solved is primitive, so the algorithm runs in polynomial time. □

Of course, a proof analogous to the one just given holds for finding a minimum weight cover by stable sets, so Theorem 2 has a similar corollary.

5. Applications to Clique Separable and Gallai Graphs

A chord (u,v) of a cycle in a graph is a pair of adjacent but nonconsecutive vertices of the cycle. Two chords (u,v) and (x,y) are said to cross if u, x, v and y are distinct and appear in this order on the cycle. In [8], Gallai studied the family of graphs defined by the property that odd cycles of length at least five always have a pair of noncrossing chords. The chordal (sometimes called "triangulated") graphs of Dirac [7], for example, satisfy this condition; these graphs, which have been studied algorithmically by Gavril [9] and Rose [17], are defined by the property that every cycle of length at least four has a chord. Gallai gave several characterizations of his graphs and showed them to be α-perfect. Suranyi [18] later gave a simpler proof of the same result. A review of Gallai's graphs, called "i-triangulated", appears in Berge [1].

In order to describe a result of Gallai's that we use, we need to define two special types of graphs. We say that a graph G is a type 1 graph if its vertices can be partitioned into two disjoint sets V_1 and V_2 such that $G(V_1)$ is a connected bipartite graph, $G(V_2)$ is a clique, and every vertex of V_1 is adjacent in G to very vertex of V_2. Here, one of V_1, V_2 may be empty. We say that G is a type 2 graph if it is a complete k-partite graph for some k.

Remark 2: Problems I - VI can be solved in polynomial time for graphs that are type 1 or 2; also, it is easy to see that these graphs are perfect.

To see that a graph G can be quickly tested for being type 1, find all vertices of degree n-1, where n is the number of vertices of G, and then check that the induced subgraph on the remaining vertices is connected and bipartite. If this induced subgraph is a stable set, a vertex of degree n-1 can be

added to it to make it connected and bipartite. Suppose that G has been found to be type 1. Then taking the vertex of maximum weight on each side of the bipartite subgraph of G together with all the vertices of degree n-1 gives a maximum weight clique for G. A maximum weight stable set can be obtained by choosing the heavier side of the bipartite subgraph of G or the vertex of degree n-1 that has maximum weight, as appropriate.

It is easy to check whether a given graph is type 2 because in type 2 graphs, nonadjacency is an equivalence relation on the vertices.

Burlet and Uhry [5] give polynomial time algorithms for solving problems II, IV, V, and VI for the family of "parity" graphs; this family includes the bipartite graphs. Using this fact, it is easy to see that these problems have polynomial time solutions for the type 1 graphs.

It is easy to check that the remaining problems concerning type 1 and type 2 graphs can be solved in polynomial time. □

A result of Gallai's that we need is given in the next Remark.

Remark 3 (Gallai [8]): A Gallai graph that is primitive must be a type 1 or type 2 graph.

Gavril [10] calls a graph clique separable if each of its primitive subgraphs is type 1 or type 2. The Gallai graphs are a strict subfamily of the clique separable graphs as Fig. 4. shows.

The clique separable graphs form quite a large family. Note, for example, that since any chordal graph that is not a clique must have a clique cutset (Hajnal and Suranyi [14]), the chordal graphs are certainly Gallai and

hence clique separable. Also, connected bipartite graphs are clique separable, as they are type 2.

It is easy to see that clique separable graphs are perfect, as they can be reconstructed from primitive perfect pieces by gluing these onto cliques of the partial reconstructions, an operation that preserves perfection.

Now we are ready to apply the results of Section 4.

Theorem 5: Let **F** be the family of clique separable graphs. Then for **F** there are polynomial time algorithms for solving problems I - VI.

Proof: To test whether a graph G belongs to **F**, build a clique cutset tree T(G), and test that each subgraph of G associated with a leaf of T(G) is type 1 or 2. All this can be done in polynomial time, by Lemma 1, Theorem 1, and Remark 2.

If G is known to be clique separable, then by Remark 2 and Theorems 2, 3, and 4 and their corollaries, there are polynomial time algorithms for solving problems II - VI. □

Theorem 5 satisfies Gavril's [10] request for polynomial time solutions to the minimum clique cover and maximum stable set problems for clique separable graphs. He solved the recognition problem and the problem of finding a minimum coloring and a maximal solution. His algorithm depends on finding clique cutsets with special properties. Algorithm 1 makes these properties unnecessary, and thus simplifies the details of his algorithm. Gavril also asked for a fast recognition algorithm for the family of graphs whose primitive members can be type 1, type 2, or "transitive orientable" ("comparability") graphs. Such an algorithm is easy to provide, as Gilmore and Hoffman

[11] and Pnueli, Lempel and Even [16] have given fast recognition algorithms for these latter graphs.

6. Gallai Graphs

Since the family of clique separable graphs strictly contains the Gallai graphs, we already have solutions to II - IV for Gallai graphs. Gavril [10] asked for a polynomial time membership test for this strict subfamily, and we are about to give such a test. We need the next Theorem in order to do this.

Theorem 5: Suppose that G is a clique separable graph, and suppose that C is an elementary cycle in G of odd length at least 5. Then either C has a pair of noncrossing chords or C has exactly one chord. In the latter case, the chord is "short", meaning that it joins two vertices that are two steps apart around the cycle.

Proof: Suppose C fails to have a pair of noncrossing chords. Then in any clique cutset tree T(G) for G, C is not a subgraph of a graph associated with a leaf, because type 1 and type 2 graphs are Gallai. Hence there must be a tree vertex (H,A) in T(G) with children (H_1,A_1), ..., (H_k,A_k), where possibly some of the (H_i,A_i)'s are leaves, such that C is not a subgraph of H_i for any i, $1 \le i \le k$. In fact, there must be i and j, $1 \le i \ne j \le k$, such that C contains a vertex of $H_i - A$ and a vertex of $H_j - A$. Of course, C must then contain at least two vertices of A and hence a chord. Cycle C cannot contain more than two vertices of A because it fails to have a pair of noncrossing chords. Hence C is a subgraph of $H_i \cup H_j$. One of $C \cap H_i$, $C \cap H_j$ must be an induced chordless odd cycle and the other, a chordless even cycle. The chordless odd cycle must have length 3, since we already know that longer odd cycles always have chords. Hence the chord given by $C \cap A$ is "short". □

Now we can give a membership test for Gallai graphs. It runs in $O(n^5)$ time.

Algorithm 3: Testing membership of a graph G in the family of Gallai graphs.

Step 1: Check whether G is clique separable, using Algorithm 2 to construct a clique cutset tree T(G) for G.

Step 2: If G is clique separable, process each nonleaf vertex of T(G) as follows. Suppose the children of (H,A) are (H_1,A_1), ...,(H_k,A_k), some of which may be leaves. Consider pairs of the form (e,v), where e is an edge of A; v is a vertex of $H_i - A$ for some i, $1 \leq i \leq k$, and is adjacent to both endpoints x and y of e. For each $j \neq i$, $1 \leq j \leq k$, construct a graph S_j as follows. Begin with $S_j = H_j - A$, but then remove all vertices adjacent to both x and y. Finally, add in x and y to S_j, putting in all induced edges except edge e.

Next, test whether there is a path from x to y in S_j. If there is, G is not Gallai: The existence of a path from x to y in S_j implies the existence of a chordless path from x to y. If this chordless path has an odd number of vertices, it forms with e a chordless odd cycle of length at least 5 in G, as x and y have no common neighbors in S_j. If the chordless path has even length, then in G it forms with the vertex v of H_i a cycle of odd length at least 5 that has only one chord.

Repeat step 2 until all nonleaf vertices have been processed. If the processing of these vertices did not yield a proof that G fails to be Gallai, then by Theorem 5, G is indeed a Gallai graph. The total time to process all the nonleaf vertices of T(G) is at most $O(n^5)$.

Burlet [3] has extensively studied Gallai graphs from an algorithmic

point of view and has independently given a recognition algorithm for them as well as algorithms for solving the optimization problems. His algorithms depend on finding clique cutsets with special properties, which are not necessary from our point of view. Burlet works with simplicial clique cutsets, meaning clique cutsets that have no edge on a chordless even length cycle. In the next theorem, we again illustrate the usefulness of the clique cutset idea as a tool for structure elucidation by showing in a different way from Burlet that these simplicial cliques exist.

Theorem 6: Suppose that G is a Gallai graph and that T(G) is a clique cutset tree for G that has been constructed subject to the following rules:

i) At each stage of the construction, choose a minimal cutset.

ii) Construct children for a tree vertex (H,A) if, and only if, H fails to be a 2-connected graph of type 1 or 2.

Then if (H,A) is a tree vertex whose children are all leaves, A is a simplicial clique of G.

Proof: Suppose by way of contradiction that (H,A) is a tree vertex whose children $(H_1,-)$, ..., $(H_k,-)$ all are leaves, but that the clique cutset A associated with vertex (H,A) contains an edge e whose endpoints x and y are consecutive vertices on a chordless even length cycle C. Then $A \cap C = e$. Let $K = A \cup C$, and note that K has no clique cutset: Such a cutset would have to separate a vertex $u \in A - e$ from a vertex $v \in C - e$. But $C \cup u$ is an odd cycle of length at least 5 and so must have another chord in addition to e, because G is Gallai. The existence of this additional chord prevents the separation of u from v by a clique cutset.

Now imagine "percolating" K through T(G). That is, observe which graphs associated with tree vertices have K as a subgraph. Since K has no clique cutset, it must be a subgraph of one of the children H_i of H. In other words, the cycle C belongs to H_i. Note that in each child H_j of H, $j \neq i$, any vertex v adjacent to both x and y must in fact belong to A. This means that if H_j, which is type 1 or type 2, contains a chordless even cycle in which x and y appear consecutively, then H_j must be a bipartite graph completely connected to the clique A - e. Consequently, there must be some H_j, $j \neq i$, in which x and y do not appear consecutively on a chordless cycle of even length: Otherwise, H itself would consist of a connected bipartite graph completely connected to the clique A - e, and H would be 2-connected. This contradicts the properties of T(G).

Let H_j be a child of H in which x and y do not appear as consecutive vertices on a chordless cycle of even length. This means that if H_j is a type 2 graph, one of x and y must be the lone vertex in its class, and so this vertex is adjacent to all other vertices of H_j. If H_j is type 1, then again, one of x and y must be adjacent to all the other vertices of H_j. Otherwise, one of x and y would be an articulation point for the bipartite piece of H_j and would form with A - e a smaller clique cutset for H, contradicting the minimality of A.

Say x is the endpoint of e that is adjacent to all other vertices of H_j. Then y is adjacent to no vertex of $H_j - A$. This is because any common neighbor of x and y in H_j would form with C and the chordless even cycle in H_i a cycle of odd length at least 5 containing only one chord. But this means that x together with A - e is a clique cutset for H, contradicting the minimality of A. □

Burlet and Fonlupt [4] have another polynomial time recognition algorithm for Gallai graphs that they obtained from their recognition algorithm for Meyniel graphs, which are defined by the property that every odd cycle of length at least five contains at least two chords. (Note that the graph in Fig. 4 shows that clique separable graphs need not be Meyniel.)

7. Conclusions

We have seen that having a quick way to find clique cutsets proves polynomial time algorithms for testing membership in families of graphs whose primitive members can be recognized in polynomial time. We have also seen that certain optimization problems can be solved quickly in these families by using the clique cutset tree approach. As an application, we pointed out that for clique separable graphs there are polynomial time algorithms for all these problems. Also, we provided a recognition algorithm for Gallai graphs which is different from the one of Burlet [3] and Burlet and Fonlupt [4].

We have seen that the clique cutset tree approach can provide a powerful tool for structure elucidation. Again, we point out that this approach is in no way restricted to perfect graphs.

We close with two questions, one in the realm of perfect graphs. First, can all perfect graphs be obtained from some collection of "primitive" perfect graphs by a sequence of perfection preserving operations? The second question is simply a more general version of a question Gavril [10], who asked for a characterization of clique separable graphs in terms of conditions on cycles. We ask for choices of "primitive" graphs giving rise to interesting families that can be defined in another way, by conditions on cycles for example.

Acknowledgement

The author wishes to thank V. Chvátal for numerous stimulating and enlightening discussions about this work. Among other things, he suggested that Theorem 4 might be true and pointed out several of the references. Also, the author wishes to thank the Computer Science Department at Cornell University for its hospitality during the academic year 1981-1982.

References

1. C. Berge, Graphs and Hypergraphs, North-Holland, Amsterdam, 1972.

2. M. Boulala and J.P. Uhry, "Polytope des indépendants d'un graphe série-parallèle", Discrete Math. **27** (1979), 225-243.

3. M. Burlet, "Etude algorithmique de certaines classes de graphes parfaits", thesis, l'Université Scientifique et Médicale de Grenoble and l'Institut National Polytechnique de Grenoble, 1981.

4. M. Burlet and J. Fonlupt, "Polynomial algorithm to recognize a Meyniel Graph", research report 303, Labatoire d'Informatique et de Mathématiques Appliquées de Grenoble, May 1982.

5. M. Burlet and J.P. Uhry, "Parity graphs", in Topics in Perfect Graphs (C. Berge and V. Chvátal, Eds.), North-Holland, to appear.

6. V. Chvátal, "On certain polytopes associated with graphs", J. Combinatorial Theory (B) **18** (1975), 138-154.

7. G.A. Dirac, "On rigid circuit graphs", Abh. Math. Sem. Univ. Hamburg **25** (1961), 71-76.

8. T. Gallai, "Graphen mit triangulierbaren ungeraden vielecken", Magyar Tud. Akad. Mat. Kutato Int. Közl. **7** (1962), 3-36.

9. F. Gavril, "Algorithms for minimum coloring, maximum clique, minimum covering by cliques and maximum independent set of a chordal graph", SIAM J. Comput. **1** (1972), 180-187.

10. F. Gavril, "Algorithms on clique separable graphs", Discrete Math. **19** (1977), 159-165.

11. P.S. Gilmore and A.J. Hoffman, "A characterization of comparability graphs and of interval graphs", Canad. J. of Math. **16** (1964), 539-548.

12. M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.

13. M. Grötschel, L. Lovász and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization", Combinatorica **1** (1981), 169-197.

14. A. Hajnal and J. Surányi, "Uber die auflösung von graphen in vollständige teilgraphen", Ann. Univ. Sc. Budapestinensis **1** (1958), 113-121.

15. L. Lovász, "Normal hypergraphs and the perfect graph conjecture", Discrete Math. **2** (1972), 253-267.

16. A. Pnueli, A. Lempel and S. Even, "Transitive orientation of graphs and identification of permutation graphs, Canad. J. of Math **23** (1971), 160-175.

17. D.J. Rose, "Triangulated graphs and the elimination process", J. Math. Anal. Appl. **32** (1970), 597-609.

18. L. Surányi, "The covering of graphs by cliques", Studia Sc. Math. Hung. **3** (1968), 345-349.

19. S.H. Whitesides, "An algorithm for finding clique cut-sets", Information Processing Letters **12** (1981), 31-32.

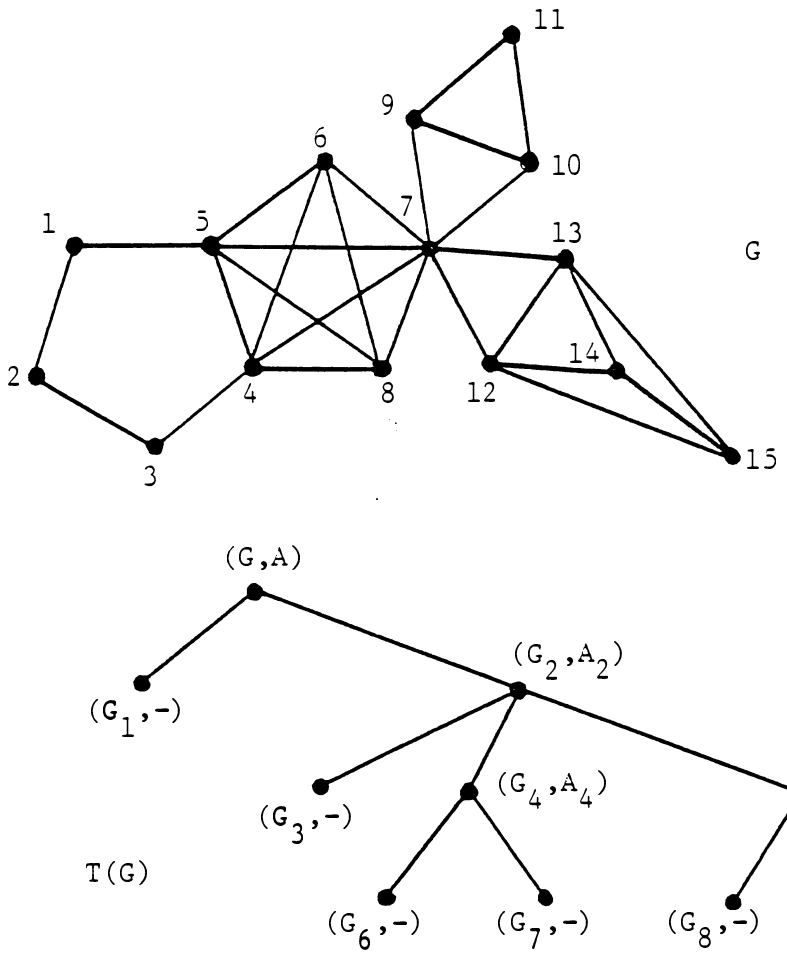

Fig. 1: A graph G and a cutset tree T(G) for G.

Here,

G = {1,...,15} and A = {4,5};

G_1 = {1,...,5}, a chordless cycle of length 5;

G_2 = {4,...,15} and A_2 = {7};

G_3 = {4,...,8}, a clique;

G_4 = {7,9,10,11} and A_4 = {9,10};

G_5 = {7,12,13,14,15} and A_5 = {12,13};

G_6 = {7,9,10}, G_7 = {9,10,11}, G_8 = {7,12,13}, and G_9 = {12,13,14,15}, where $G_6,\ldots,G_9$ are cliques. (We have denoted induced subgraphs of G by their vertex sets.)

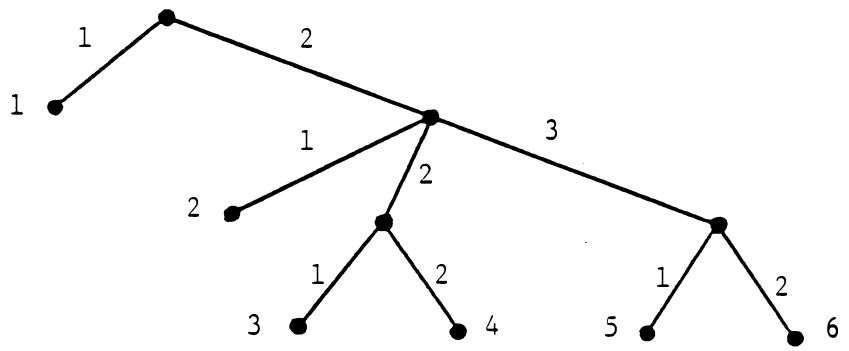

Fig. 2: A numbering for the leaves of the tree T(G) of Fig. 1. The numbers on the edges represent the ordering assigned to children in step 2i) of Algorithm 2. The numbers on the leaves represent the "gluing" order assigned to the leaves by step 2iii).

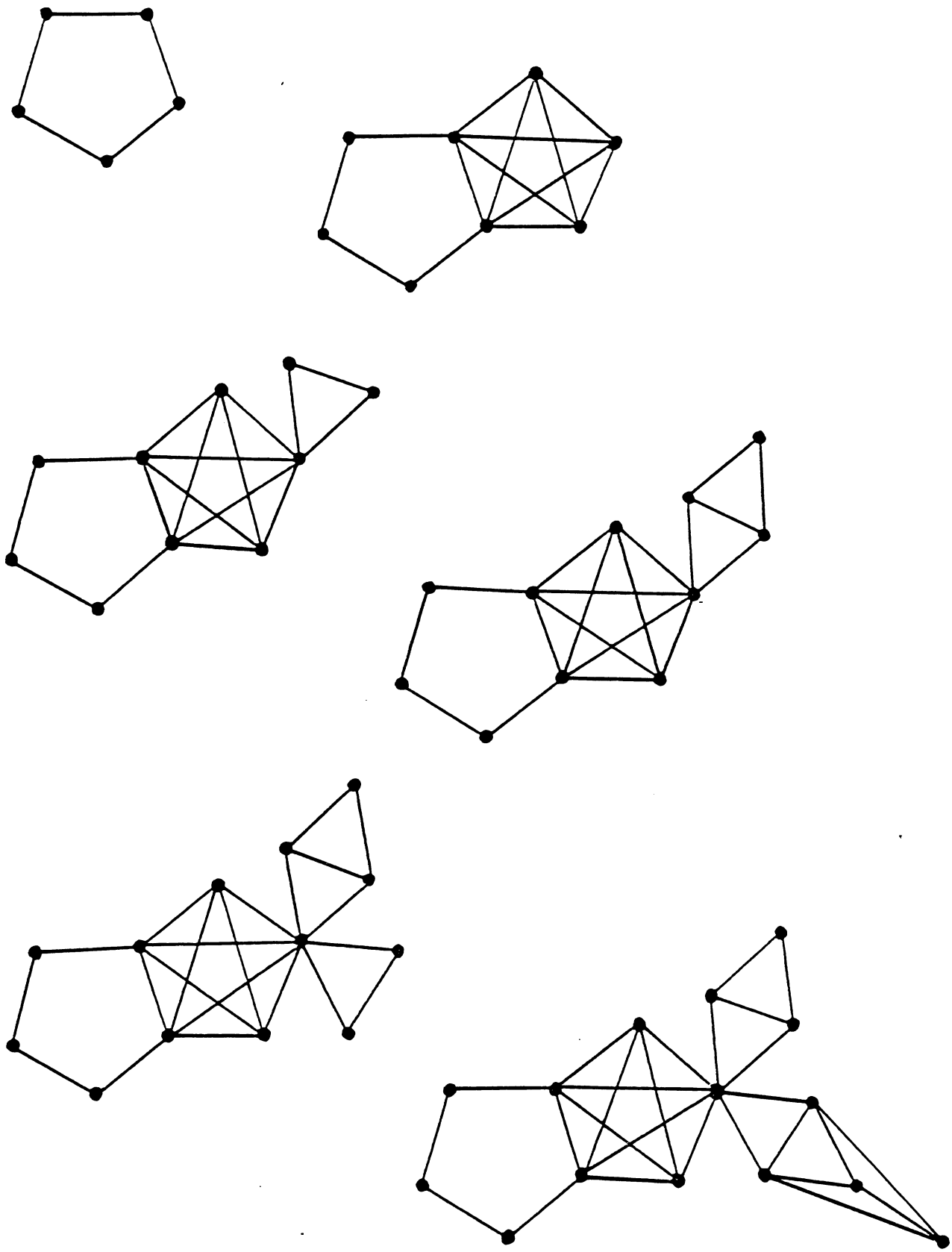

Fig. 3: Reconstructing the graph G of Fig. 1
according to the gluing order shown in Fig. 2.

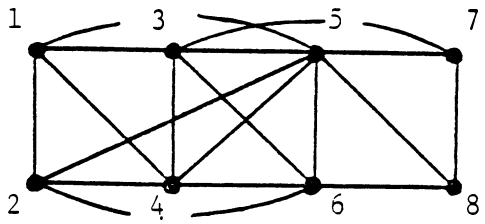

Fig. 4: A clique separable graph G that is not Gallai (i-triangulated). Vertices 3, 4, 6, 8, and 7 constitute an odd cycle that has only (3,6) as a chord. Nevertheless, $A = \{3,5,6\}$ is a clique cutset, where the children of G produced by A are $G_1 = \{1,\ldots,6\}$ and $G_2 = \{3,5,6,7,8\}$. Both G_1 and G_2 are complete 3-partite graphs.