Discrete Mathematics 55 (1985) 221–232
North-Holland

DECOMPOSITION BY CLIQUE SEPARATORS

Robert E. TARJAN

AT & T, Bell Laboratories, Murray Hill, NJ 07974, U.S.A.

Received 17 August 1983
Revised 9 August 1984

We consider the problem of decomposing a graph by means of clique separators, by which we mean finding cliques (complete graphs) whose removal disconnects the graph. We give an $O(nm)$-time algorithm for finding a decomposition of an n-vertex, m-edge graph. We describe how such a decomposition can be used in divide-and-conquer algorithms for various graph problems, such as graph coloring and finding maximum independent sets. We survey classes of graphs for which such divide-and-conquer algorithms are especially useful.

0. Notation

Throughout this paper $G = (V, E)$ is an undirected graph with vertex set V and edge set E. We denote an edge by $\{v, w\} \in E$, with $v, w \in V$. We denote by n the number of vertices and by m the number of edges in G. We assume G is connected and $n \geq 2$; thus $m \geq 1$. If X is a subset of the vertices, $G(X)$ is the subgraph of G induced by X; if $G(V - X)$ is disconnected, X is a *separator*. By a path we mean a simple path (no vertex is repeated); by a cycle we mean a simple cycle (no vertex is repeated except the first, which occurs only as the first and last). A *clique* is a set of pairwise adjacent vertices.

1. Introduction

A technique used in many graph algorithms is divide-and-conquer [1]. To apply this technique, we decompose the input graph into a hierarchy of components. Then we solve the problem on each of the components at the bottom of the hierarchy, which we call *atoms*, and gradually piece together the solutions on larger and larger components, until finally computing a solution for the entire graph.

Usually the decomposition that supports divide-and-conquer is based on finding separators of small size. Examples are the following:

(1) Decomposition of trees and more generally arbitrary graphs via separators of size one; the atoms are the biconnected components or blocks [23].

(2) Decomposition of series-parallel [22] and arbitrary graphs via separators of size two; the atoms are the triconnected components [13].

(3) Decomposition of planar graphs via separators of size $O(\sqrt{n})$ [16].

In this paper we focus not on the size but on the structure of the separators. In particular, we study decomposition by separators that are cliques, which we call *clique* separators.

Suppose G has a clique separator C. Let A, B, C be a vertex partition such that no vertex in A is adjacent to a vertex in B. Then we can decompose G into components $G' = G(A \cup C)$ and $G'' = G(B \cup C)$, separated by C. By decomposing G' and G'' in the same way and repeating until no further decomposition is possible, we decompose G into a collection of atoms, each a subgraph of G containing no clique separator. The atoms fit together in a hierarchy to form G. We can represent this hierarchy by a binary tree; each external node represents an atom and each internal node represents a clique separator. We call such a tree a *binary decomposition tree.* (See Fig. 1.)

The binary decomposition tree of a graph is far from unique; indeed, by varying the decomposition order we can obtain different sets of atoms. (See Fig. 2.) Gavril [10] used an alternative definition of decomposition that allows simultaneous decomposition into more than two components by a single separator, but this is not enough to guarantee uniqueness. We leave open the problem of defining a unique decomposition; none of our results depends on uniqueness.

The remainder of the paper consists of three sections. In Section 2 we present an $O(nm)$-time algorithm for finding a decomposition by clique separators. In Section 3 we describe how such a decomposition can be used to solve graph problems efficiently. We consider four NP-complete problems; for each, we show

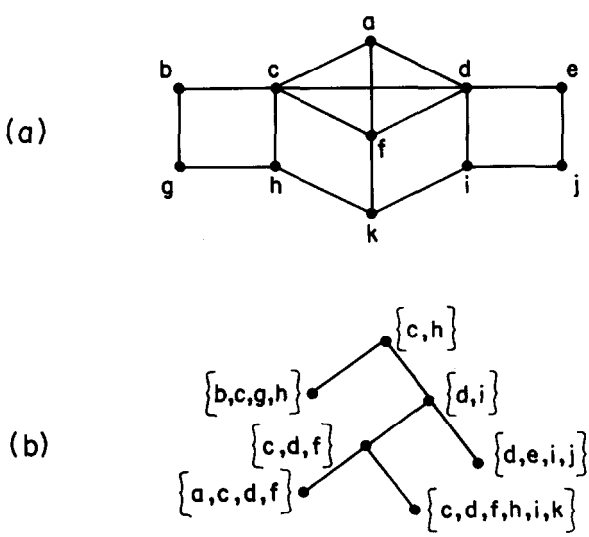

Fig. 1. A graph and its binary decomposition tree. (a) Graph. (b) Decomposition tree. The labels on the internal nodes are the vertex sets of the corresponding cliques, the labels on the external nodes are the vertex sets of the corresponding subgraphs.

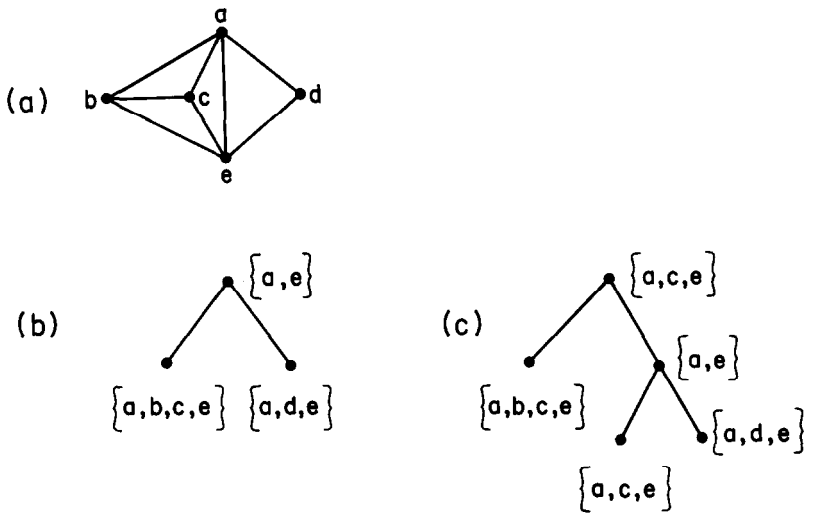

Fig. 2. A graph with different decompositions. (a) Graph. (b) One decomposition. (c) Another decomposition.

how to combine exact or approximate solutions on the atoms to give an exact or approximate solution on the entire graph. The problems we consider are those of minimizing the fill-in caused by Gaussian elimination, finding a maximum clique, graph coloring, and finding a maximum independent set. In Section 4 we survey classes of graphs for which decomposition by clique separators is especially useful.

2. A decomposition algorithm

One way to decompose a graph by clique separators is to use an algorithm devised by Whitesides [25] for finding a clique separator. Although she claimed an $O(n^3)$ running time for her algorithm, an analysis gives a bound of $O(nm)$. By applying this algorithm repeatedly, we can find a decomposition. The time required depends on the number of atoms. As noted by Gavril [10], a crude upper bound on this number is $\binom{n}{2} - m$, since the number of unordered vertex pairs v, w, such that v and w are in the same component and $\{v, w\}$ is not an edge, decreases by at least one with each decomposition step. Thus the time to find a decomposition with this method is $O(n^3m)$.

The theory of clique separators is intimately related to the theory of elimination orderings [18–20]. We can obtain a faster decomposition algorithm (and a better decomposition) by using a minimal elimination ordering to generate clique separators. To understand the method we need some terminology. A *chord* of a cycle is an edge between two non-consecutive vertices on the cycle. A graph G is *chordal* if every cycle of length four or more has a chord. An *elimination ordering*

π is a numbering of the vertices of G from 1 to n. The *fill-in* F_π caused by the ordering π is the set of edges defined as follows:

$$F_\pi = \{\{v, w\} \mid v \neq w, \{v, w\} \notin E, \text{ and there is a path } v = v_1, v_2, \ldots, v_k = w \text{ in } G \text{ such that } \pi(v_i) < \min\{\pi(v), \pi(w)\} \text{ for } i = 2, \ldots, k-1\}.$$

An elimination ordering π is *perfect* if $F_\pi = \emptyset$, *minimum* if $|F_\pi|$ is minimum over all possible orderings, and *minimal* if there is no ordering σ such that $F_\sigma \subset F_\pi$, where '$\subset$' denotes strict containment. The graph $G_\pi = (V, E \cup F_\pi)$ is the *fill-in graph* for π.

Elimination orderings arise in the study of Gaussian elimination on sparse symmetric matrices [18, 19]. We shall need the following properties of such orderings:

Theorem A ([20]). *Any ordering π is a perfect elimination ordering of G_π.*

Theorem B ([3, 7, 20]). *G has a perfect elimination ordering if and only if G is chordal.*

The definition of chordality implies that any induced subgraph of a chordal graph is chordal. Our first result relates minimal orderings and decompositions by clique separators.

Lemma 1. *Let π be a minimal ordering. Let C be a clique separator of G. Then no edge in F_π joins vertices in different connected components of $G(V-C)$.*

Proof. Let $X_1, X_2, \ldots, X_k$ be the vertex sets of the connected components of $G(V-C)$. Form F' from F_π by discarding all edges joining vertices in different sets X_i. We claim $G' = (V, E \cup F')$ is chordal. Consider any cycle in G' of length four or more. If the cycle lies entirely in $G(C \cup X_i)$ for some i, it has a chord F_π, which is also in F'. If the cycle contains vertices from two or more sets X_i, it must contain two non-consecutive vertices that are both in C, and thus it has a chord in E. This proves the claim and the lemma, since the minimality of F_π implies $F' = F_\pi$. □

Theorem 1. *Let π be a minimal ordering. For any decomposition by clique separators, every edge $\{v, w\} \in F_\pi$ is such that a unique atom contains both v and w.*

Proof. Immediate from Lemma 1. □

Now we are ready to state our decomposition algorithm. The algorithm consists of two steps. First, we find a minimal ordering π and compute $C(v) = \{w \mid \pi(w) > \pi(v) \text{ and } \{v, w\} \in E \cup F(\pi)\}$ for each vertex v. Then we repeat the

following step for each vertex v in increasing order with respect to π (see Fig. 3):

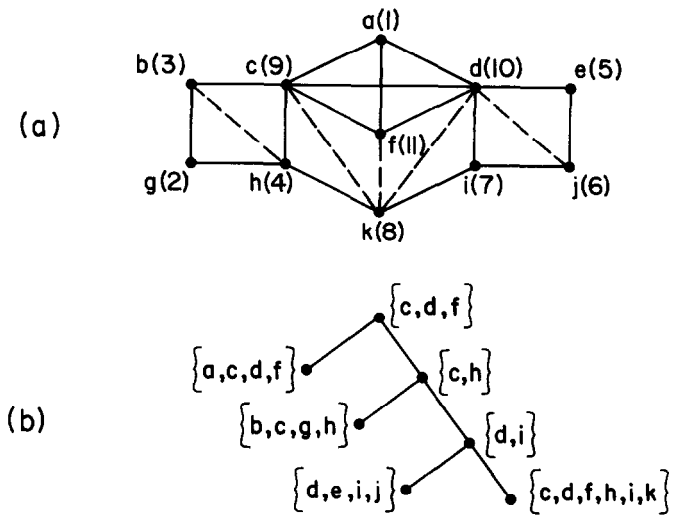

Fig. 3. Application of the decomposition algorithm. (a) Minimal ordering for the graph of Fig. 1. Fill-in edges are dashed. (b) Tree generated by the algorithm. The decomposition step succeeds for vertices a, b and j, producing clique separators $\{c, d, f\}$, $\{c, d\}$ and $\{d, i\}$, respectively.

Decomposition step. Let A be the vertex set of the connected component of $G(V-C(v))$ containing v, and let $B = V-(C(v)\cup A)$. If $C(v)$ is a clique in G and $B\neq\emptyset$, decompose G into $G' = G(A\cup C(v))$ and $G'' = G(B\cup C(v))$, separated by $C(v)$. Replace G by G''.

We shall call a decomposition step *successful* if it finds a clique separator. The following lemma is the heart of the correctness proof of the algorithm.

Lemma 2. *If G contains a clique separator, some decomposition step will be successful.*

Proof. Suppose G has a clique separator C. Choose C minimal, i.e., such that no proper subset is a separator. Let A_1 and A_2 be the vertex sets of two of the connected components of $G(V-C)$. By the minimality of C, every vertex in C is adjacent to at least one vertex in A_i for $i=1,2$. Let x and y be the maximum vertices in A_1 and A_2, respectively (with respect to π). Suppose there is a vertex $z\in C$ such that $\pi(z)<\min\{\pi(x), \pi(y)\}$. There is a path p in $G_\pi(A_1)$ from x to a vertex adjacent to z. Theorem A and the definition of fill-in imply that if u, v, w are vertices such that $\pi(v)<\min\{\pi(u), \pi(w)\}$ and $\{u, v\}, \{v, w\}\in E\cup F_\pi$, then $\{u, w\}\in E\cup F_\pi$. It follows that there is a path $x = x_1, x_2, \ldots, x_j = z$ such that $x_i\in A_1$ and $\pi(x_i)>\pi(x_{i+1})$ for $i=1,\ldots,j-1$. Similarly there is a path $y = y_1, y_2, \ldots, y_k = z$ such that $y_i\in A_2$ and $\pi(y_i)>\pi(y_{i+1})$ for $i=1,\ldots,k-1$. Again by Theorem A and the definition of fill-in, $\{x, y\}\in F_\pi$, contradicting Lemma 1.

This contradiction implies that for every vertex $z \in C$, $\pi(z) > \min\{\pi(x), \pi(y)\}$. Suppose without loss of generality that $\pi(x) < \pi(y)$. An argument like the one in the previous paragraph shows that $C(x) = C$, where $C(x)$ is as defined in the decomposition algorithm. Thus the algorithm will succeed either on x or on a previously processed vertex. □

Theorem 2. *The decomposition algorithm is correct.*

Proof. Every successful decomposition step produces a clique separator, and if G has a clique separator the algorithm will find one. Consider the first successful decomposition step, which decomposes G into $G' = G(A \cup C(v))$ and $G'' = (B \cup C(v))$, separated by $C(v)$. The definitions of A and $C(v)$ imply that every vertex in A is less than every vertex in $C(v)$. The ordering π imposes orderings π' on G' and π'' on G''. By Lemma 1, $F_{\pi'}$ is the subset of F_π in $G_\pi(A \cup C)$ and $F_{\pi''}$ is the subset of F_π in $G_\pi(B \cup C)$. For any vertex $x \in A$, $C'(x) = C(x)$, where $C'(x)$ is defined in G' with respect to ordering π'. For any vertex $x \in C(v)$, $C'(x)$ is the set of vertices in $C(v)$ greater than x. Thus if the decomposition algorithm is run on G' with ordering π', it will not find a clique separator, which means by Lemma 2 that G' has no clique separator. For any vertex $y \in B \cup C(v)$, $C''(y) = C(y)$, where $C''(y)$ is defined in G'' with respect to π''. Thus the continuation of the algorithm behaves just as if it had been started on G'' with ordering π''. An induction on the number of successful decomposition steps proves that the algorithm is correct. □

We can estimate the running time of this algorithm as follows. Finding a minimal ordering takes $O(nm)$ time using either the algorithm of Rose, Tarjan and Lueker [20] or the algorithm of Ohtsuki, Cheung and Fujisawa [17]. Computing the sets $C(v)$ is essentially a matter of computing F_π, which takes $O(|E \cup F_\pi|) = O(n^2)$ time using the algorithm of Rose, Tarjan and Lueker [20]. Applying the decomposition step to a single vertex takes $O(m)$ time. The total time is thus $O(nm)$.

The algorithm produces a decomposition with at most $n-1$ atoms. (A connected graph of two vertices is not decomposable.) We can improve the algorithm slightly if instead of decomposing using the separator $C(v)$ at each step, we use $C'(v) = \{w \in C(v) \mid w$ is adjacent to at least one vertex in $V - (C(v) \cup A)\}$ and avoid applying the decomposition step to any vertex in $C(v) - C'(v)$. (Such vertices are in G' but not in G''.) Although this will not improve the asymptotic worst-case behavior of the algorithm, it may result in fewer atoms and smaller separators.

The decomposition tree produced by the algorithm is as skewed as possible: the internal nodes lie on one path. We shall use this to simplify two of the algorithms we present in the next section.

3. Applications to NP-complete problems

In this section we describe how a decomposition by clique separators can be used to speed up the solution of hard graph problems. We consider four NP-complete problems. For each, we show how to combine exact or approximate solutions on the atoms (or subgraphs of the atoms) to give an exact or approximate solution on the entire graph.

3.1. *Minimum fill-in*

Consider the problem of finding a minimum elimination ordering, or at least an ordering that makes the fill-in small. Deciding whether there is an ordering that produces k or fewer fill-in edges, where k is a problem parameter, is NP-complete [26]. However, suppose we have a way to find good orderings on the atoms. Then we can compute a good ordering on the entire graph, as follows. Let the atoms be $G_i = (V_i, E_i)$ with ordering π_i for $i = 1, \ldots, k$. First, we compute the fill-in F_i on G_i produced by π_i. Since $G_i' = (V_i, E_i \cup F_i)$ is chordal by Theorems A and B, so is $G' = (V, E \cup \bigcup_{i=1}^{k} F_i)$. Next, we compute a perfect ordering π on G'. The fill-in produced by π on G is a subset of $\bigcup_{i=1}^{k} F_i$. Thus π is minimum if π_i is minimum for all i, and π is a good approximation to a minimum ordering if all the π are close to minimum. The time for this computation is $O(m')$, where $m' = |E \cup \bigcup_{i=1}^{k} F_i|$, using the fill-in and perfect ordering algorithms of Rose, Tarjan and Lueker [20], not counting the time to find good orderings on the atoms.

3.2. *Maximum clique*

Consider the problem of finding a maximum-weight clique, where each vertex has a real-valued weight. Deciding whether there is a clique of weight at least w, where w is a problem parameter, is NP-complete, even if every vertex has the same weight [18]. However, suppose we can find maximum or at least large-weight cliques in the atoms. Any clique in G cannot be separated, and thus appears in at least one atom. Therefore the maximum-weight clique among the cliques found in the atoms is maximum in G if the cliques in the atoms are maximum, or a good approximation to a maximum clique if the cliques in the atoms are good approximations.

3.3. *Graph coloring*

Consider the problem of coloring the vertices of a graph with a minimum number of colors so that no two adjacent vertices have the same color. Deciding whether there is a coloring that uses k colors or less is NP-complete, even for $k = 3$ [8]. However, suppose we can find colorings with k or fewer colors on the atoms. We can combine these colorings to give a k-coloring on the entire graph,

using the following recursive method. Let A, B, C be a vertex partition such that C is a clique, no edge joins a vertex in A and a vertex in B, and $G(A \cup C)$ is an atom. We color $G(B \cup C)$ by applying the algorithm recursively. Then we extend the coloring to G by renaming the colors in the known k-coloring of $A \cup C$ so that they match the colors on C in $G(B \cup C)$. The time required for this method is proportional to the total number of vertices in all the atoms, which is $O(n^2)$, not counting the time to color the atoms. The method uses the fact that the decomposition algorithm described in Section 2 produces at least one atom in each decomposition step.

3.4. *Maximum independent sets*

Consider the problem of finding a maximum-weight independent set (set of pairwise non-adjacent vertices), where each vertex has a real-valued weight. Deciding whether there is an independent set of weight at least w, where w is a problem parameter, is NP-complete, even if every vertex has the same weight [8]. However, if we can find maximum-weight independent sets on certain subgraphs of the atoms, then we can find a maximum-weight independent set for the entire graph. We use the following recursive method. Let A, B, C be a vertex partition such that C is a clique, no edge joins a vertex in A and a vertex in B, and $G(A \cup C)$ is an atom. We denote by $\text{wt}(I)$ the total weight of vertex set I.

Step 1. For each vertex $v \in C$, determine a maximum-weight independent set $I(v)$ in $G(A - \text{adj}(v))$, where $\text{adj}(v)$ is the set of vertices adjacent to v. Determine a maximum-weight independent set I' in $G(A)$.

Step 2. For each vertex $v \in C$, redefine the weight of v to be $\text{wt}(\{v\}) + \text{wt}(I(v)) - \text{wt}(I')$. Find a maximum-weight independent set I'' in $G'' = G(B \cup C)$ with respect to the new weights.

Step 3. Define $I = I(v) \cup I''$ if $v \in I'' \cap C$, $I = I' \cup I''$ if $I'' \cap C = \emptyset$.

The correctness of this method is obvious. The time to find a maximum independent set in the entire graph is $O(n^2)$, not counting the time necessary to find maximum independent sets in subgraphs of the atoms. We must solve $O(n)$ independent set problems per atoms for a total of $O(n^2)$ subproblems.

The maximum independent set problem is equivalent to the maximum clique problem on the complementary graph, and thus the method of Section 3.2 can be used if the complementary graph is easy to separate by cliques.

4. Classes of graphs decomposable by clique separators

There are several classes of graphs for which decomposition by clique separators is especially useful. We shall mention three; the reader can undoubtedly find others.

4.1. *Chordal graphs*

Dirac [3] proved that in a chordal graph every minimal separator is a clique. (See also [18, 19].) It follows that the chordal graphs are exactly those graphs whose atoms are cliques. The algorithms of Section 3 allow us to find maximum-weight cliques, minimum colorings, and maximum-weight independent sets in chordal graphs, since these problems are easy to solve on cliques. Algorithms for these problems on chordal graphs were given by Gavril [9] for all but maximum-weight independent set and by Frank [6] for the latter problem.

4.2. *Clique separable graphs*

Gavril [10] defined the class of *clique separable graphs*. A graph is clique separable if all of its atoms are of type one or type two, defined as follows. A graph $G = (V, E)$ is of *type one* if V can be partitioned into V_1 and V_2 such that $G(V_1)$ is bipartite, $G(V_2)$ is a clique, and if $v \in V_1$ and $w \in V_2$ then $\{v, w\} \in E$. A graph $G = (V, E)$ is of *type two* if it is complete k-partite for some k, i.e. V can be partitioned into $V_1, V_2, \ldots, V_k$ such that $\{v, w\} \in E$ if and only if $v \in V_i$ and $w \in V_j$ for some $i \neq j$. Gavril gave a polynomial-time algorithm for recognizing clique separable graphs; such an algorithm follows immediately from our decomposition algorithm in Section 2. Gavril also gave algorithms for finding minimum colorings and maximum cliques in clique separable graphs; his algorithms are the specializations of the methods in Sections 3.2 and 3.3 to these graphs.

Concerning minimum elimination orderings and maximum-weight independent sets in clique separable graphs, we have the following results. The problem of finding a minimum elimination ordering on a bipartite graph is NP-complete, as the problem for general graphs can be reduced to the bipartite case by replacing every edge by a suitably large collection of parallel paths of length two. Thus the minimum ordering problem on clique separable graphs is NP-complete. Finding a maximum-weight independent set in a type one graph can be reduced to $O(n)$ instances of the same problem on a bipartite graph. On a bipartite graph, the problem is solvable in polynomial time using network-flow techniques [5]. A maximum-weight independent set in a type two graph is easy to find, since it is just a maximum-weight set of vertices in one of the parts. Combining these observations with the algorithm of Section 3.4 we can find a maximum-weight independent set in a clique separable graph in polynomial time, solving a problem left open by Gavril.

4.3. *EPT graphs*

Golumbic and Jamison [11, 12] defined the class of *EPT graphs* (edge intersection graphs of paths in a tree). An EPT graph is a graph whose vertices can be represented as paths in a tree such that two vertices are adjacent if and only if the corresponding paths overlap in at least one edge.

In order to understand the role of clique separators in EPT graphs, we need to

define two other classes of graphs. A *star* is a tree having one vertex adjacent to all the rest. A graph is a *line graph* if its vertices correspond to the edges of a multigraph (graph with multiple edges) such that two vertices in the multigraph are adjacent if and only if the corresponding edges in the original graph share a common vertex.

The following results are easy to verify (see [11, 12]). A graph is a line graph if and only if it is the EPT graph of a star. A clique in an EPT graph corresponds either to a set of paths containing a common edge (an *edge clique*) or to a set of paths each containing two out of three selected edges intersecting at a vertex (a *claw clique*). The atoms of any EPT graph are line graphs, although not every graph whose atoms are line graphs is EPT: indeed, there is a polynomial-time algorithm to recognize line graphs [15], but recognizing EPT graphs is NP-complete [11, 12].

The characterization of the cliques of an EPT graph gives a polynomial-time algorithm for finding maximum-weight cliques in EPT graphs [11, 12]. Coloring the vertices of a line graph is equivalent to coloring the edges of a multigraph. Edge-coloring a multigraph is NP-complete [14], but a theorem of Shannon [21] gives an approximation algorithm that uses no more than 1.5 times the minimum number of colors. This approximation algorithm extends to EPT graphs by the method of Section 3.3. Other edge-coloring theorems (see [2]), such as that of Vizing [24], give similar approximation algorithms for coloring EPT graphs. Finding a maximum-weight independent set in a line graph is just the maximum weighted matching problem, which has a polynomial-time algorithm [4]. This algorithm extends to finding maximum-weight independent sets in EPT graphs by the results of Section 3.4. The complexity of the minimum fill-in problem on line graphs, and hence on EPT graphs, is open.

Note added in proof

C.L. Monma and V.K. Wei [27] have raised the question of whether the algorithm of Section 2 can be modified so that only maximal cliques are used for decomposition. (A clique is *maximal* if it is not properly contained in another clique.) The answer is yes. We compute a minimal ordering π and the set $C(v)$ for each vertex v as before. Then we repeat the following step for each vertex v in increasing order with respect to π:

Decomposition step. Let A be the vertex set of the connected component of $G(V-C(v))$ containing v, and let $B = V-C(v)-A$. There are two cases; if both apply, either may be selected.

(i) If $C(v)\cup\{v\}$ is a maximal clique in G, $A\neq\{v\}$, and $B\neq\emptyset$, decompose G into $G' = G(A\cup C(v))$ and $G'' = G(B\cup C(v)\cup\{v\})$, separated by $C(v)\cup\{v\}$. Replace G by G''.

(ii) If $B'\subseteq B$ is a (possibly empty) set such that $B'\cup C(v)$ is a maximal clique in

G and $B - B' \neq \emptyset$, decompose G into $G' = G(A \cup C(v) \cup B')$ and $G'' = G(B \cup C(v))$, separated by $B' \cup C(v)$. Replace G by G''.

An extension of the argument in Section 2 shows that this algorithm is correct. Its running time bound is the same as that of the original algorithm, namely $O(nm)$.

Acknowledgment

My thanks to Marty Golumbic, whose lecture on EPT graphs stimulated this research.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).
[2] C. Berge, Graphs and Hypergraphs (North-Holland, Amsterdam, 1973).
[3] G.A. Dirac, On rigid circuit graphs, Abh. Math. Sem. Univ. Hamburg 25 (1961) 71–76.
[4] J. Edmonds, Matching and a polyhedron with 0-1 vertices, J. Res. Nat. Bur. Standards 69B (1965) 125–130.
[5] L.R. Ford, Jr. and D.R. Fulkerson, Flows in Networks (Princeton University Press, Princeton, NJ, 1962).
[6] A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, Proc. Fifth British Combinatorial Conference 1975, Congressus Numeratium XV (Utilitas Mathematica, Winnipeg, 1976) 211–226.
[7] D.R. Fulkerson and O. Gross, Incidence matrices and interval graphs, Pacific J. Math. 15 (1965) 835–855.
[8] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (Freeman, San Francisco, CA, 1979).
[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] M.C. Golumbic and R.E. Jamison, The edge intersection graphs of paths in a tree, J. Combin. Theory Ser. B, to appear.
[12] M.C. Golumbic and R.E. Jamison, Edge and vertex intersection of paths in trees, Discrete Math. 55 (1985) 151–159.
[13] J.E. Hopcroft and R.E. Tarjan, Dividing a graph into triconnected components, SIAM J. Comput. 2 (1973) 135–158.
[14] I. Holyer, The NP-completeness of edge coloring, SIAM J. Comput. 10 (1981) 718–720.
[15] P.G.H. Lehot, An optimal algorithm to detect a line graph and output its root graph, J. ACM 21 (1974) 569–575.
[16] R.J. Lipton and R.E. Tarjan, A separator theorem for planar graphs, SIAM, J. Appl. Math. 36 (1979) 177–189.
[17] T. Ohtsuki, L.K. Cheung and T. Fujisawa, Minimal triangulation of a graph and optimal pivoting order in a sparse matrix, J. Math. Anal. Appl. 54 (1976) 622–633.
[18] D.J. Rose, Triangulated graphs and the elimination process, J. Math. Anal. Appl. 32 (1970) 597–609.
[19] D.J. Rose, a graph-theoretic study of the numerical solution of sparse positive definitive systems of linear equations, in: R. Read, ed., Graph Theory and Computing (Academic Press, New York, 1973) 183–217.
[20] D.J. Rose, R.E. Tarjan and G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, SIAM J. Comput. 5 (1976) 266–283.

[21] C.E. Shannon, A theorem on coloring the lines of a network, J. Math. Phys. 28 (1949) 148–151.
[22] K. Takamizawa, T. Nishizeki and N. Saito, Linear-time computability of combinatorial problems on series-parallel graphs, J. ACM 29 (1982) 623–641.
[23] R.E. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (1972) 146–160.
[24] V.G. Vizing, On an estimate of the chromatic class of a p-graph (in Russian), Diskret. Analiz. 3 (1964) 25–30.
[25] S.H. Whitesides, An algorithm for finding clique cut-sets, Inform. Proc. Letters 12 (1981) 31–32.
[26]. M. Yannakakis, Computing the minimum fill-in is NP-complete, SIAM J. Algebraic Discrete Meth. 2 (1981) 77–79.
[27] C.L. Monma and V.K. Wei, Intersection graphs of paths in a tree, Bell Communications Research, Morristown, NJ (1985).