Discrete Mathematics 55 (1985) 245–254
North-Holland

COMPOSITIONS FOR PERFECT GRAPHS

G. CORNUÉJOLS*

Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.

W.H. CUNNINGHAM**

Carleton University, Ottawa, Ont. K1S 5B6, Canada

Received 10 October 1983
Revised 10 December 1984

In this paper we introduce a new graph composition, called 2-amalgam, and we prove that the 2-amalgam of perfect graphs is perfect. This composition generalizes many of the operations known to preserve perfection, such as the clique identification, substitution, join and amalgam operations. We give polynominal-time algorithms to determined whether a general graph is decomposable with respect to the 2-amalgam or amalgam operations.

1. Introduction

An operation which, given two graphs G_1 and G_2, constructs a third graph G will be called a *composition*. We write $G = G_1 * G_2$. Conversely, a given graph G can be $*$-*decomposed* if there exist graphs G_1 and G_2 such that $G = G_1 * G_2$ and each of G_1 and G_2 has fewer nodes than G.

Perfect graphs were introduced by Berge [1] as those graphs for which, in every node-induced subgraph, the size of a largest clique is equal to the chromatic number.

In a very nice paper Burlet and Fonlupt [4] defined a composition of graphs, called amalgam, and showed how to use it to characterize in polynomial time a class of perfect graphs known as Meyniel graphs [10]. Their main results are

(i) The amalgam of two Meyniel graphs is a Meyniel graph.

(ii) Conversely any Meyniel graph can be amalgam-decomposed in polynomial time into 'basic' Meyniel graphs.

(iii) 'Basic' Meyniel graphs can be recognized in polynomial time.

It is natural to try a similar approach for the class of all perfect graphs since, at present, there is no polynomial-time algorithm to recognize perfect graphs.

Several compositions of graphs are known to preserve perfection: union, clique identification [2], graph substitution [9], join [3, 6, 8], amalgam [4]. In this paper

* Supported in part by NSF grant ECS-8205425 and an Alexander Von Humboldt fellowship, while at the Institute für Operations Research, Universität Bonn.

** Supported in part by SFB21(DFG), Institut für Operations Research, Universität Bonn, and by NSERC of Canada.

we describe a new composition of graphs, called the 2-amalgam, which generalizes and unifies all these compositions. In fact this operation, together with complementation, encompasses many of the operations previously known to preserve perfection.

We also give a polynomial-time algorithm to 2-amalgam decompose a general graph or show that no such decomposition exists. For a graph with n nodes and m edges the complexity of the algorithm is $O(m^2n^2)$. To find an amalgam-decomposition, the complexity reduces to $O(mn^2)$. Algorithms of complexity $O(n^3)$ have already appeared for finding a clique cutset or a join-decomposition in a graph, see [11] and [7] respectively.

2. A graph composition which preserves perfection

Given the graphs G_1 and G_2, we define the composition Φ_{ik} as follows.

For $j = 1, 2$, consider a clique of size $i+k$ in G_j with nodes $\{v_1^j, \ldots, v_i^j\} \cup K_j$, and let U_j be the remaining set of nodes in G_j. Assume that no node of U_j is adjacent to more than one node v_h^j. Furthermore each node of U_j which is adjacent to v_h^j for some $h = 1, \ldots, i$ is also adjacent to all the nodes in K_j.

The composed graph $G = G_1 * G_2$ is obtained by identifying the cliques K_1 and K_2 and, for each $h = 1, \ldots, i$ deleting v_h^1 and v_h^2 and joining every neighbor of v_h^1 to every neighbor of v_h^2.

Φ_{00} is the union of G_1 and G_2; Φ_{0k} is a clique identification; Φ_{10} is the join of G_1 and G_2; Φ_{1k} is the amalgam. Φ_{2k} is called the 2-*amalgam* of G_1 and G_2 (see Fig. 1). The 2-*join* is the special case of the 2-amalgam where $k = 0$.

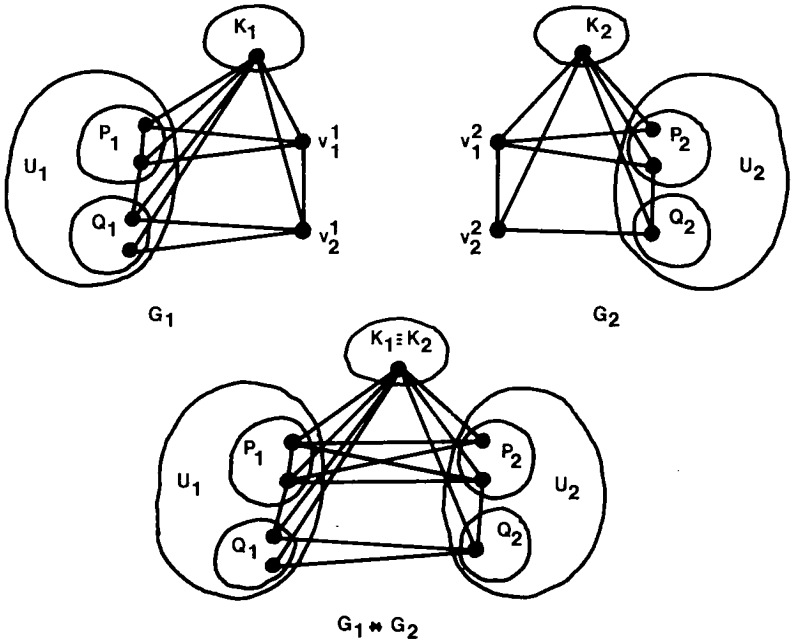

Fig. 1. An example of the 2-amalgam composition.

We shall prove that the 2-amalgam preserves perfection. The proof will use the concept of *node duplication* in a graph. The operation of duplicating a node u consists of adding a new node joined to u and to its neighbors. Node duplication preserves perfection [9].

For $j = 1, 2$, let $V_j = U_j \cup K_j$ and let ω_j be the size of the largest clique in $G[V_j]$ (the graph induced by the node set V_j), p_j the size of the largest clique in $G[P_j]$ where P_j is the set of nodes of U_j which are adjacent to v_1^j, and q_j the size of the largest clique in $G[Q_j]$ where Q_j is the set of nodes of U_j which are adjacent to v_2^j. Given a coloring of the nodes of a graph, $C(S)$ denotes the set of colors appearing on node set S.

Lemma 1. *Let G_j^* be obtained from G_j by duplicating node v_2^j into node u_2^j, and let H_j be obtained from G_j^* by deleting the edge $v_1^j u_2^j$. If G_j is perfect, then H_j is also perfect.*

Proof. Assume not. Let H_j' be a minimal imperfect subgraph of H_j and let G_j' be the subgraph of G_j^* induced by the same node set, say W_j. Since G_j' is perfect and H_j' is minimal imperfect, W_j must constain the nodes v_1^j and u_2^j. In addition, the chromatic numbers of H_j' and G_j' must be equal and their clique numbers must differ by one. Therefore there is a unique clique of maximum size in G_j' and this clique contains both nodes v_1^j and u_2^j. Since the only nodes of G_j' adjacent to both v_1^j and u_2^j are those of $W_j \cap (K_j \cup \{v_2^j\})$, the unique largest clique of G_j' is $W_j \cap (K_j \cup \{v_1^j, v_2^j, u_2^j\})$. The uniqueness requires that $W_j \cap Q_j = \emptyset$ (otherwise a node in this intersection could be used in place of v_1^j to form a clique of the same size.) Therefore $W_j \cap (K_j \cup \{v_1^j\})$ is a clique cutset of the graph H_j'. This contradicts the assumption that H_j' is minimally imperfect since clique identification preserves perfection. □

Lemma 2. *If G_j is a perfect graph and ω is an integer, $\omega \geq \omega_j$, then there exists a coloring of $G[V_j]$ with at most ω colors such that $|C(P_j)| = p_j$, $|C(Q_j)| = q_j$ and $|C(P_j) \cap C(Q_j)| = \max(0, p_j + q_j + k - \omega)$.*

Proof. Consider the graph H_j defined in Lemma 1. Duplicate $\omega - (p_j + k)$ times the node v_1^j, duplicate $\min(\omega - (q_j + k), p_j)$ times the node v_2^j and duplicate $\max(0, \omega - (p_j + q_j + k))$ times the node u_2^j. (Duplicating zero times a node means deleting this node.) Duplication preserves perfection, so the new graph, say H, is still perfect. Note that the nodes v_1^j are duplicated just enough times so that they all belong to a clique of size ω. Similarly for the nodes v_2^j and the nodes u_2^j. So the size of the largest clique in H is ω. Now consider a minimum coloring of H. It contains ω colors. Since K_j and the duplicates of v_1^j from a clique of size $\omega - p$ we must have $C(P_j) = p_j$. Similarly K_j and the duplicates of v_2^j and u_2^j form a clique of size $\omega - q_j$, so we must have $C(Q_j) = q_j$. Finally consider the clique K formed by K_j and the duplicates of v_1^j and v_2^j. If $\omega \geq r_j + q_j + k$, then this clique has cardinality ω and therefore we must have $C(P_j) \cap C(Q_j) = \emptyset$. If $\omega \leq p_j + q_j + k$, then the clique K has cardinality $k + [\omega - (p_j + k)] + [\omega - (q_j + k)] = 2\omega - (p_j + q_j + k)$ and therefore

$p_i + q_i + k - \omega$ colors do not appear in the clique K. The colors that do not appear in K_i and the duplicates of v_1^i are exactly the colors that appear in P_i. Similarly the colors that do not appear in K_i and the duplicates of v_2^i are exactly those appearing in Q_i. So the $p_i + q_i + k - \omega$ colors which do not appear in K are exactly those that appear in both P_i and Q_i. This completes the proof of the lemma. □

Theorem 1. *The 2-amalgam preserves perfection.*

Proof. Assume that G_1 and G_2 are perfect and let $G = G_1 * G_2$. The size of the largest clique in G is $\omega = \max(\omega_1, \omega_2, p_1 + p_2 + k, q_1 + q_2 + k)$. We will construct a coloring of G with ω colors. This will be sufficient since any node induced subgraph of G is obtained as the 2-amalgam of the corresponding node-induced subgraphs of G_1 and G_2.

Consider colorings of $G[V_1]$ and $G[V_2]$ with the properties described in Lemma 2. In $G[V_1]$, label the p_1 colors that appear in P_1 by the integers 1 through p_1 and the q_1 colors that appear in Q_1 by the integers $\omega - k - q_1 + 1$ through $\omega - k$. In $G[V_2]$, label the p_2 colors that appear in P_2 by the integers $\omega - k - p_2 + 1$ through $\omega - k$ and the q_2 colors that appear in Q_2 by the integers 1 through q_2. In both $G[V_1]$ and $G[V_2]$, label the k colors that appear in K_1 and K_2 by the integers $\omega - k + 1$ through ω.

If the coloring is not valid, there must be a common color in P_1 and P_2 or in Q_1 and Q_2. Then either $p_1 \geq \omega - k - p_2 + 1$ or $q_2 \geq \omega - k - q_1 + 1$, contradicting the definition of ω. □

Note that Theorem 1 does not generalize to i-amalgams Φ_{ik} for $i \geq 3$. For example Fig. 2 shows that the 3-join $\Phi_{3,0}$ of two perfect graphs can contain a 7-hole.

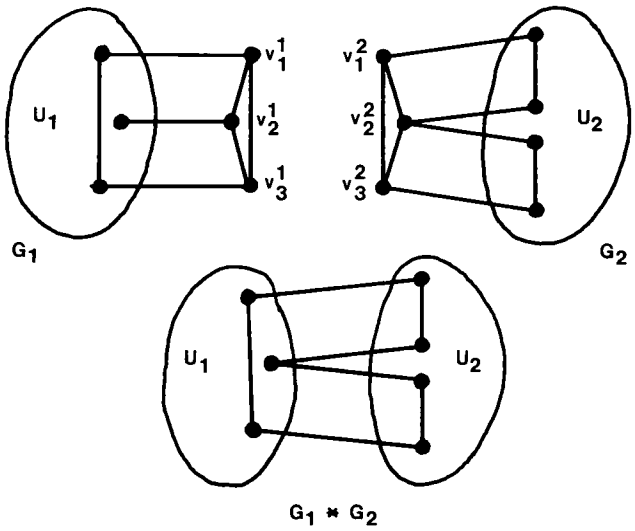

Fig. 2. The 3-join composition does not preserve perfection.

3. Decomposition algorithms

Burlet and Fonlupt [4] presented an efficient algorithm either to show that a given graph G is an amalgam of smaller graphs or to show that G is not a Meyniel graph. Here we describe the first efficient algorithm to determine whether an arbitrary graph is an amalgam of smaller graphs. We also show how similar ideas can be applied to the recognition of i-amalgams, $i \geq 2$.

First, we mention previous work on algorithms for some of the more special compositions. A number of algorithms have been proposed for recognizing substitution-decomposability; the first polynomial-time one seems to be in [5]. Finding a 'clique cutset', if one exists, is equivalent to determining whether a graph arises from smaller graphs by clique identification. There is an elegant and efficient algorithm for this problem [11]. Finally, a polynomial-time algorithm for recognizing join-decomposability (which includes by a simple construction recognition of substitution-decomposability) was given in [7].

In this section we let V, E denote the node-set and edge-set of G, and we put $n = |V|$, $m = |E|$. We assume for convenience that G is connected. Given a partition (A_1, C, A_2) of V into three sets, let B_1 denote $\{u \in A_1 : uv \in E \text{ for some } v \in A_2\}$, and similarly for B_2.

We say that (A_1, C, A_2) is an (*amalgam*) *split* of G if:

(i) $|A_1| \geq 2 \leq |A_2|$;

(ii) $uv \in E$ whenever $u, v \in C$, $u \neq v$;

(iii) $uv \in E$ whenever $u \in C$, $v \in B_1 \cup B_2$;

(iv) $uv \in E$ whenever $u \in B_1$, $v \in B_2$.

It is easy to see that G is amalgam-decomposable if and only if it admits a split (A_1, C, A_2) as above. If (A_1, C, A_2) has the property that one (and thus both) of B_1, B_2 is empty, then C is a clique cutset. We may suppose that the $O(nm)$ algorithm [11] for finding clique cutsets has already been applied, so we restrict attention here to the existence of splits (A_1, C, A_2) for which B_1 and B_2 are non-empty.

The algorithm for finding a split of G, or determining that there is none, uses ideas introduced in [7] for the case $C = \emptyset$. We give an $O(n^2)$ algorithm to determine for a fixed edge $xy \in E$, whether there is a split (A_1, C, A_2) for which $x \in A_1$, $y \in A_2$. Such an algorithm can be used to provide an $O(n^2 m)$ algorithm to decide whether G has a split. (In the case $C = \emptyset$, the resulting algorithm is $O(n^3)$, because it is enough to run the basic algorithm for each edge xy of some spanning tree of G.)

Henceforth, we assume that $|V| \geq 4$, and we deal with a fixed edge xy of G. A preliminary step is to find a node $z \neq x$, y such that no split (A_1, C, A_2) with $x \in A_1$, $y \in A_2$ satisfies $z \in C$. There is a simple procedure to find such a node, if G is not complete. (Of course, if G *is* complete, then $(A_1, \emptyset, A_2)$ is a split whenever $|A_1| \geq 2 \leq |A_2|$.) Choose two non-adjacent nodes u, v. If either is not a common neighbor of x and y, then it is certainly an acceptable choice for z. In the

alternative case, either of u, v may be chosen to be z. (If not, we would have $u, v \in C$, or $u \in B_1 \cup B_2$ and $v \in C$, or $u \in C$ and $v \in B_1 \cup B_2$; each of these implies that $uv \in E$.) Now any split (A_1, C, A_2) for which $x \in A_1$ and $y \in A_2$ satifies $z \in A_1$ or $z \in A_2$, so it will be enough to give an $O(n^2)$ algorithm to solve the following problem. (It will be necessary to use that algorithm twice, once with the roles of x and y interchanged.)

(1) *Problem.* Find a split (A_1, C, A_2) satisfying $z, x \in A_1$, $y \in A_2$, or determine that there is none.

Consider a partition (S, K, T) of V having the following property:

(2) $x, z \in S$, $y \in T$, and xv, $yv \in E$ for all $v \in K$; moreover, for any split (A_1, C, A_2) with $x, z \in A_1$ and $y \in A_2$, we have $S \subseteq A_1$ and $A_2 \subseteq T$.

Initially, putting $S = \{x, z\}$ and $K = \emptyset$ determines a partition satisfying (2). On the other hand, if (S, K, T) satisfies (2) with $T = \{y\}$, then we know that there is no positive solution to (1). The algorithm maintains (S, K, T) satisfying (2) and, at each step, either recognizes that (S, K, T) is the desired split or finds an element which can be moved from T to S, from T to K, or from K to S. The rules for moving elements of V are simple, and we describe and justify them now. Henceforth, (S, K, T) always denotes a partition of V, so specifying two of these sets determines the third. Throughout, it is assumed that (S, K, T), (A_1, C, A_2) are as in (2).

Rule 1. If $u \in S$, $v \in T$, $uy \in E$, $xv \in E$, $uv \notin E$, then v can be added to S.

Justification. Since $uy \in E$, $u \in S$ we have $u \in B_1$. If $v \in C$, then $uv \in E$, a contradiction. If $v \in A_2$, then, since $xv \in E$, $v \in B_2$ and so $uv \in E$, a contradiction. Hence $v \in A_1$, as required.

Rule 2. If $u \in S$, $v \in T$, $uv \in E$, $xv \notin E$, then v can be added to S.

Justification. Clearly $x \in B_1$, and, if $v \in A_2$, then $v \in B_2$. Thus $v \in C$ or $v \in A_2$ would imply $xv \in E$, a contradiction, so $v \in A_1$.

Rule 3. If $u \in S$, $v \in T$, $uv \in E$, $uy \notin E$, then v can be added to K if xv, $yv \in E$, and otherwise v can be added to S.

Justification. Since $uy \notin E$, $u \in S$, we must have $u \in A_1 \setminus B_1$. Therefore, since $uv \in E$, we must have $v \in A_1 \cup C$. However, $v \in C$ implies vx, $vy \in E$, so if one of these fails v can be added to S, and otherwise v can be added to K.

Rule 4. If $u \in S$, $v \in K$, $uy \in E$, $uv \notin E$, then v can be added to S.

Justification. Since $uy \in E$, $u \in S$, we must have $u \in B_1$, so $v \in C$ would imply $uv \in E$.

Rule 5. If $u \in K$, $v \in T$, $xv \in E$, $uv \notin E$, then v can be added to S.

Justification. Since $u \in K$, we have $uy \in E$, so $u \in B_1 \cup C$. Since $xv \in E$, $v \in A_1 \cup C \cup B_2$, but $v \in C$ or $v \in B_2$ would imply $uv \in E$, a contradiction.

Rule 6. If $u, v \in K$, $u \neq v$ and $uv \notin E$, then u and v can be added to S.

Justification. Since $u, v \in K$, we have $uy, vy \in E$, so $u, v \in B_1 \cup C$. But if one or both of u, v are in C, then $uv \in E$, a contradiction.

Proposition. *Suppose, beginning with* $S = \{x, z\}$ *and* $K = \emptyset$, *Rules* 1 *through* 6 *are used repeatedly until no further application is possible. If* $|T| \geq 2$, *then* (S, K, T) *is the split required in* (1), *and otherwise no such split exists.*

Proof. The second part of the claim, that $|T| < 2$ implies that no such split exists, is immediate from the fact that the initial choice of (S, K, T) satisfies (2) and that Rules 1–6 preserve (2). Now suppose that $|T| \geq 2$. We must show that $A_1 = S$, $C = K$, $A_2 = T$ satisfies (i)–(iv). Of course, (i) is satisfied, and (ii) follows from the fact that Rule 6 cannot be applied. Now suppose that $u \in C$ and $v \in B_1 \cup B_2$. Since u can enter K only by Rule 3, we have $ux, uy \in E$. Now if $v \in B_1$, then since Rule 4 cannot be applied, we have $uv \in E$. Similarly, if $v \in B_2$, then since Rule 5 cannot be applied we have $uv \in E$. Therefore, (iii) is satisfied. Finally, suppose that $u \in B_1$, $v \in B_2$. By the definition of B_1, B_2 there exist $p \in A_1$, $q \in A_2$ with $uq, pv \in E$. Since Rule 2 cannot be applied, we have $xv \in E$ and, since Rule 3 cannot be applied, we have $uy \in E$. Then, since Rule 1 cannot be applied, we have $uv \in E$. Thus (iv) is proved, so (S, K, T) is a split. □

It is now clear that our suggested algorithm is correct and that it will run in polynomial time. However, we claim that it can be implemented to run in time $O(n^2)$ for each choice of x, y. The preliminary step which finds z is clearly $O(n^2)$. All of Rules 1 to 6 are stated in terms of (some or all of) nodes u, v, x, y. Given the adjacency lists for each of these nodes in characteristic vector form and (S, K, T) represented by a $(0, 1, -1)$-vector, we can decide whether one of Rules 1 to 6 can be applied, and make any necessary change to (S, K, T) in constant time. To enable the algorithm to perform correctly with only $O(n^2)$ such operations, we process the nodes in a special order. Suppose that $u \in S$, and we want to check for applications of Rules 1 to 4. Any $v \notin S$ which cannot be added to S as a result of such an application, cannot later be added to S, using the current u. That is, we can check for all such applications, for a fixed u, at one time.

We maintain a list L_1 of elements of S to be scanned, and a list L_2 of elements

of K to be scanned. Initially, $L_1 = \{x, z\}$, and $L_2 = \emptyset$. Each time an element is added to S it is added to L_1, and each time an element is added to K it is added to L_2. When an element is scanned it is deleted from its list. Scanning an element of L_1 means asking it to play the role of u in Rules 1 to 4. Scanning an element of K means asking it to play the role of u in Rules 5 and 6. The algorithm terminates when L_1 and L_2 are empty. Clearly, every node is scanned at most twice, and each scanning operation requires $O(n)$ time, so we obtain the desired $O(n^2)$ bound. Since we must run this algorithm for every choice of x, y, we have an $O(n^2m)$ algorithm to find an amalgam split.

Now we consider the recognition of 2-amalgam decomposability. In this case we require that the partition (A_1, C, A_2) satisfy (ii), (iii), and (i'), (iv') below.

(i') $|A_1| \geq 3 \leq |A_2|$;

(iv') There exists a partition $\{B_{i1}, B_{i2}\}$ of B_i, $i = 1$ and 2, such that if $u \in B_{1j}$, $v \in B_{2k}$ then $uv \in E$ if and only if $j = k$.

The method for finding, if possible, such a partition is a natural extension of that used for the amalgam. (As usual, we assume first that G is not decomposable with respect to any of the simpler decompositions.) Where $x_1y_1, x_2y_2 \in E$ and $x_1y_2, x_2y_1 \notin E$, we try to find (A_1, C, A_2) as above for which $x_i \in B_{1i}$, $y_i \in B_{2i}$, $i = 1$ and 2. (Necessarily, x_1, y_1, x_2, y_2 must be distinct.) Again, it is ncessary to find a node $z \neq x_1$, y_1, x_2, y_2 such that $z \notin C$ for any such partition. Any node which is not a common neighbor of x_1, y_1, x_2, y_2 will do, as will any node which is not adjacent to some common neighbor. If no such z exists, G has at most 5 nodes (because otherwise $\{x_1, y_1, x_2, y_2\}$ and its complement yield a joint decomposition), and so G is not 2-amalgam decomposable. Any partition (A_1, C, A_2) of the kind required must satisfy either $z \in A_1$ or $z \in A_2$, so it will be enough to describe an algorithm to find (A_1, C, A_2) such that $x_1, x_2, z \in A_1$ and $y_1, y_2 \in A_2$.

We begin with $S = \{x_1, x_2, z\}$ and $C = \emptyset$, and apply a set of rules similar to those for the amalgam. Each of Rules 1 to 6 have analogues for the present situation. As examples, we give two of these analogues.

Rule 1'. If $u \in S$, $v \in T$, $uv \notin E$ and for some i, $uy_i \in E$, $x_iv \in E$, then v can be added to S.

Rule 5'. If $u \in K$, $v \in T$, $uv \notin E$ and, for some i, $x_iv \in E$, then v can be added to S.

We also need two new rules, both based on the requirement that $B_{i1} \cap B_{i2} = \emptyset$ for $i = 1$ and 2.

Rule 7'. If $v \in T$ and $x_1v, x_2v \in E$, then v can be added to K if $vy_1, vy_2 \in E$, and otherwise v can be added to S.

Rule 8'. If $u \in S$ and $uy_1, uy_2 \in E$, then stop; there can be no 2-amalgam split (A_1, C, A_2) with $S \subseteq A$, $y_1, y_2 \in A_2$.

So the algorithm can terminate by using Rule 8′ as well as by encountering $T = \{y_1, y_2\}$. Similar implementation techniques to the ones described before, can be used to obtain an $O(n^2)$ time bound for this algorithm. Since there are $O(m^2)$ possible choices for x_1, y_1, x_2, y_2, we obtain an $O(n^2m^2)$ algorithm for the recognition of 2-amalgam decomposability. Similarly, there is an $O(n^2m^i)$ algorithm for i-amalgam decomposability. For the special case of 2-join decomposability, on can obtain a bound of $O(n^3m)$. The reason is that, for any spanning tree T of G, at least one edge of T must go from A_1 to A_2. Such an edge can be chosen for x_1y_1. Thus we have only $n-1$, not m, choices for x_1y_1, leading to the improved bound.

It is interesting to remark that the algorithms presented in this paper, as well as those in [11, 7] for clique cutsets and join-decomposability, either prove that no decomposition exists or find a decomposition into two smaller graphs one of which is irreducible. Therefore, at most n applications of these algorithms are needed to decompose a graph G into irreducible factors.

Finally, we point out two facts. First, the 2-join of two imperfect graphs can be perfect. Second, a minimal imperfect graph can be obtained as a 2-join. To avoid these difficulties, one may want to require that the two graphs being composed be isomorphic to induced subgraphs of the composition graph. This requirement is automatically satisfied by clique-identification, the join and the amalgam compositions. For the 2-amalgam (and the 2-join) it is satisfied provided that there is at least one edge joining some node of B_{i1} to some node of B_{i2} for $i = 1$ and 2. The question arises whether 2-amalgam decomposability with this additional requirement can be recognized efficiently. In fact, it can, and with the same efficiency as for the ordinary 2-amalgam. Namely, we can restrict the choice of x_1, y_1, x_2, y_2 to the case where $x_1x_2, y_1y_2 \in E$.

Acknowledgment

We are grateful to Bob Bixby and László Lovász for pointing out and correcting an error in the original proof of Theorem 1.

References

[1] C. Berge, Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise stan sind, Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur, Reihe (1961) 114.

[2] C. Berge, Graphs and Hypergraphs (North-Holland, Amsterdam, 1972).

[3] R.E. Bixby, A composition for perfect graphs, in: C. Berge and V. Chvátal, eds., Topics on Perfect Graphs, Ann. Discrete Math. 21 (North-Holland, Amsterdam, 1984) 221–224.

[4] M. Burlet and J. Fonlupt, Polynomial algorithm to recognize a Meyniel graph, in: C. Berge and V. Chvátal, eds., Topics on Perfect Graphs, Ann. Discrete Math. 21 (North-Holland, Amsterdam, 1984) 225–252.

[5] D.D. Cowan, L.O. James and R.G. Stanton, Graph decomposition for undirected graphs, Proc. 3rd Southeastern Conference on Combinatorics (Utilitas Math., Winnipeg, Canada, 1972).
[6] W.H. Cunningham, A combinatorial decomposition theory, Thesis, University of Waterloo, Ontario, Canada (1973).
[7] W.H. Cunningham, Decomposition of directed graphs, SIAM. J. Algebraic Discrete Methods 3 (1982) 214–228.
[8] W.H. Cunningham and J. Edmonds, A combinatorial decomposition theory, Canad. J. Math. 32 (1980) 734–765.
[9] L. Lovász, Normal hypergraphs and the perfect graph conjecture, Discrete Math. 2 (1972) 253–267.
[10] H. Meyniel, On the perfect graph conjecture, Discrete Math. 16 (1976) 339–342.
[11] S. Whitesides, An algorithm for finding clique cutsets, Inform. Process. Lett. 12 (1981) 31–32.