ELSEVIER Discrete Applied Mathematics 89 (1998) 59–73

DISCRETE APPLIED MATHEMATICS

The complexity of some problems related to GRAPH 3-COLORABILITY

Andreas Brandstädt*, Van Bang Le, Thomas Szymczak

Fachbereich Informatik, Universität Rostock, Rostock, Germany

Received 23 December 1997; received in revised form 16 July 1998; accepted 10 August 1998

Abstract

It is well-known that the GRAPH 3-COLORABILITY problem, deciding whether a given graph has a stable set whose deletion results in a bipartite graph, is NP-complete. We prove the following related theorems: It is NP-complete to decide whether a graph has a stable set whose deletion results in (1) a tree or (2) a trivially perfect graph, and there is a polynomial algorithm to decide if a given graph has a stable set whose deletion results in (3) the complement of a bipartite graph, (4) a split graph or (5) a threshold graph. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Graph partitions; Graph coloring; Special graph classes; NP-completeness; 3SAT, 2SAT

1. Introduction and results

Let k be a positive integer. A graph is *k-colorable* if its vertex set can be partitioned into k pairwise disjoint stable sets. Karp [12] proved that deciding whether a graph admits a k-coloring is NP-complete for $k \geqslant 3$. The problem "Is a given graph 3-colorable?" has been discussed for many special graph classes; among them planar graphs with maximum degree four [7, 6], triangle-free graphs [13], and triangle-free graphs with maximum degree four [14], and very recently 4-regular Hamiltonian graphs [17]. It turned out that GRAPH 3-COLORABILITY remains NP-complete for these graph classes.

Here we shall discuss some variants of GRAPH 3-COLORABILITY. Our discussion is motivated by the proof of the NP-completeness of GRAPH 3-COLORABILITY given by Garey et al. [7]. A *bipartite graph* is one whose vertex set can be partitioned into two disjoint stable sets. With this notion, GRAPH 3-COLORABILITY can be restated as follows: "Given a graph G, does G have a stable set S such that $G - S$ is a bipartite graph?". Now, as pointed out by Monien [16], it can be derived from the proof of Garey et al. [7] that the following holds:

* Corresponding author. E-mail: ab@informatik.uni-rostock.de.

Theorem 1 (Garey et al. [7]). *It is NP-complete to decide whether a given graph G has a stable set S such that $G - S$ is a forest.*

Notice that every bipartite graph G has a stable set S such that $G - S$ is a forest. The main result of this paper is:

Theorem 2. *It is NP-complete to decide whether a given bipartite graph G with maximum degree four has a stable set S such that $G - S$ is a tree.*

We now are going to give other related theorems. A graph is a *split graph* if its vertex set can partitioned into a clique and a stable set. A typical subclass of split graphs is the class of *threshold graphs* (graphs without chordless path P_4 and cycle C_4 with four vertices, and no complement of a C_4). Graphs without chordless P_4 and C_4 are called *trivially perfect graphs*. For more information on all these graph classes, see [3].

Theorem 3. *It is NP-complete to decide whether a given graph G has a stable set S such that $G - S$ is a trivially perfect graph.*

In the following theorems, m and $\overline{m}$ denote the number of edges of the given graph G and (of its complement) $\overline{G}$, respectively.

Theorem 4. *There is an $\mathcal{O}((n+\overline{m})^2)$ time algorithm to decide whether a given graph has a stable set S such that $G - S$ is the complement of a bipartite graph.*

Theorem 5. *There is an $\mathcal{O}((n+m)^2)$ time algorithm to decide whether a given graph has a stable set S such that $G - S$ is a split graph.*

Theorem 6. *There is a polynomial time algorithm to decide whether a given graph G has a stable set S such that $G - S$ is a threshold graph.*

The time bound in Theorem 6 is roughly $\mathcal{O}(n^8)$, and we do not try to improve this time bound in this paper.

A graph G is called *perfect* if, for each of its induced subgraphs H, H is $\omega(H)$-colorable, where $\omega(H)$ denotes the largest number of pairwise adjacent vertices in H. Grötschel et al. [8, 9] showed that Graph k-colorability can be solved in polynomial time when considering perfect graphs. Since bipartite graphs are perfect, Theorem 2 means that the question "Does G have a stable set S such that $G-S$ is a tree?" remains NP-complete for perfect graphs.

Many other related results and problems have been discussed in the literature and can be represented Table 1:

Problem 1 (Stable-π). Given a graph G, does G have a stable set S such that $G - S$ has the graph property π?

Table 1
Complexity status of STABLE-π for some property π

Property π	In general	On perfect graphs
Bipartite	NP-c [5–7]	P [8, 9]
Co-bipartite	P [1], [this paper]	P
Forest	NP-c [7]	?
Maximum degree $\leqslant 1$	NP-c [15, p. 170]	?
Tree	NP-c [this paper]	NP-c [this paper]
Split	P[1], [this paper]	P
Trivially perfect	NP-c [this paper]	?
Threshold	P [this paper]	P [this paper]
K_3-free	NP-c [4]	P [8, 9]
P_4-free	NP-c [11]	NP-c [11]

The complexity status of STABLE-π for some property π is summarized in Table 1. There, "?" means that the corresponding problem seems to be open.

In the next two sections, given a graph G and a graph property π, we call a stable set S of G *good* if S can be extended to a stable set S' such that $G-S'$ has the property π. If S is a good stable set, we shall identify S with a possible extension S' of S.

2. Proof of Theorem 2

Schaefer proved in [18] that the following problem is NP-complete.

Problem 2 (1-IN-3 3SAT). Given a collection $\mathscr{C}$ of m clauses over the set V of n Boolean variables such that each clause has exactly three literals, is there a truth assignment satisfying $\mathscr{C}$ such that each clause in $\mathscr{C}$ has exactly one true literal?

By considering complementation, the following problem is also NP-complete.

Problem 3 (2-IN-3 3SAT). Given a collection $\mathscr{C}$ of m clauses over the set V of n Boolean variables such that each clause has exactly three literals, is there a truth assignment satisfying $\mathscr{C}$ such that each clause in $\mathscr{C}$ has exactly two true literals?

We shall reduce 2-IN-3 3SAT to our problem (which is clearly in NP). Let $\mathscr{C}=\{C_1,\dots,C_m\}$ be a collection of m clauses with variable set $V=\{v_1,\dots,v_n\}$ such that every clause of $\mathscr{C}$ contains exactly three literals, $C_i=\{c_1^{(i)},c_2^{(i)},c_3^{(i)}\}$, $i=1,\dots,m$, where each literal $c_j^{(i)}$ is either v_k or $\overline{v_k}$ for some suitable k.

In the following we construct a graph $G:=G(\mathscr{C})$ such that $\mathscr{C}\in$ 2-IN-3 3SAT if and only if the graph G has a good stable set.

For a better understanding we first prove the NP-completeness for bipartite graphs.

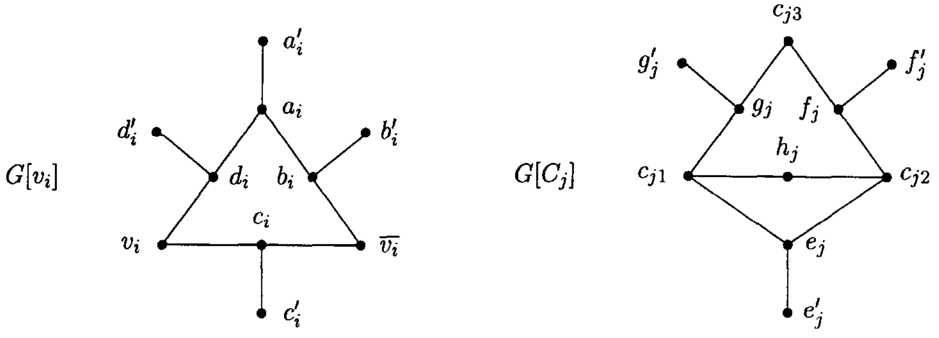


Fig. 1. The graphs $G[v_i]$ and $G[C_j]$.

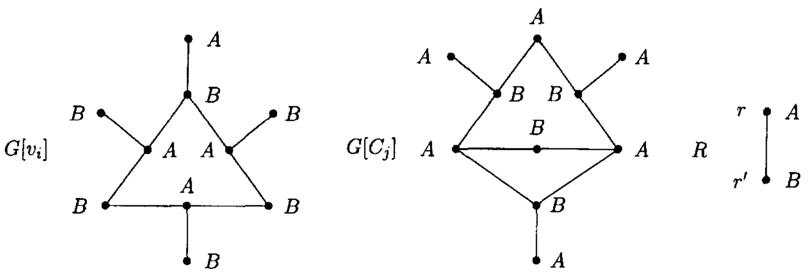


Fig. 2. Bipartition of the vertices of G.

Lemma 1. *The problem* Stable tree *is NP-complete for bipartite graphs.*

Proof. For every variable v_i we consider the graph $G[v_i]$ shown in Fig. 1 (left). For each $j \in \{1,\dots,m\}$ let $G[C_j]$ be the graph shown in Fig. 1 (right). Finally we take a graph $R = (\{r,r'\},\{rr'\})$.

Now, our graph G is obtained from $G[v_i]$, $i=1,\dots,n$, $G[C_j]$, $j=1,\dots,m$ and R by adding the following edges:

$$
\begin{array}{ll}
c_j^{(i)}c_{ij} & (i=1,\dots,m,\ j=1,2,3),\\
a_ir & (i=1,\dots,n),\\
e_jr & (j=1,\dots,m).
\end{array}
$$

Note that G is a bipartite graph: A bipartition of G is given by A, B (see also Fig. 2)

$A := \{a'_i,b_i,c_i,d_i\colon\ 1 \leqslant i \leqslant n\} \cup \{e'_j,f'_j,g'_j,c_{j1},c_{j2},c_{j3}\colon\ 1 \leqslant j \leqslant m\} \cup \{r\}$,
$B := \{a_i,b'_i,c'_i,d'_i,v_i,\overline{v_i}\colon\ 1 \leqslant i \leqslant n\} \cup \{e_j,f_j,g_j,h_j\colon\ 1 \leqslant j \leqslant m\} \cup \{r'\}$.

Assume that G has a stable set S such that $T := V(G) \backslash S$ induces a tree in G.

Claim 1. *If a vertex v in G has a neighbor w with degree one then $v \in T$. In particular a_i, b_i, c_i, d_i, e_j, f_j, g_j, $r \in T$ $(i=1,\dots,n,\ j=1,\dots,m)$.*

Proof. If $v \in S$ then w would be an isolated vertex in $G(T)$. □

Claim 2. $|\{v_i, \overline{v_i}\} \cap S| = 1$ *for all* $i \in \{1, \dots, n\}$.

Proof. If v_i, $\overline{v_i} \in T$ the vertices a_i, b_i, $\overline{v_i}$, c_i, v_i, d_i would induce a cycle in $G(T)$. If v_i, $\overline{v_i} \in S$ then there exists no path in $G(T)$ from c_i to a_i, i.e. $G(T)$ is not connected. □

Claim 3. $|\{c_{j1}, c_{j2}, c_{j3}\} \cap S| = 1$ *for all* $j \in \{1, \dots, m\}$.

Proof. Since e_j, f_j, $g_j \in T$ not all vertices in $\{c_{j1}, c_{j2}, c_{j3}\}$ can be in T since otherwise $G(T)$ contains a cycle. Suppose that two vertices in $\{c_{j1}, c_{j2}, c_{j3}\}$ are in S. If c_{j1}, $c_{j2} \in S$ then h_j is isolated in $G(T)$. If c_{j1}, $c_{j3} \in S$ then $G(T)$ contains no path connecting g_j and r. The case c_{j2}, $c_{j3} \in S$ is handled similarly. □

Claim 4. *For every* $i \in \{1, \dots, n\}$ *the graph* $G[v_i] - S$ *is a tree.*

Proof. This follows from Claims 1 and 2. □

Claim 5. *For every* $j \in \{1, \dots, m\}$ *the graph* $G[C_j] - S$ *is a tree.*

Proof. This follows from Claims 1 and 3. □

Claim 6. *For all* $i \in \{1, \dots, m\}$, $j \in \{1, 2, 3\}$ $c_j^{(i)} \in S \Leftrightarrow c_{ij} \in T$ *holds.*

Proof. The direction "$\Rightarrow$" is clear. The other direction immediately follows from Claims 4 and 5 since a_i, e_j, $r \in T$, a_ir, $e_jr \in E(G)$ and $G(T)$ contains no cycle. □

Now consider the following truth assignment

$$b(v_i) := \begin{cases} 1, & v_i \in S, \\ 0, & v_i \in T. \end{cases}$$

By Claims 3 and 6 in each clause exactly two literals are fulfilled by b.

Let b be a truth assignment for $\mathscr{C}$ such that in each clause exactly two literals are fulfilled.

We build S as follows:
(S1) If $b(v_i) = 1$ put v_i in S, otherwise put $\overline{v_i}$ in S.
(S2) If $b(c_j^{(i)}) = 0$ put c_{ij} in S.
(S3) If $b(c_1^{(j)}) = b(c_2^{(j)}) = 1$ put h_j in S.

Claim 7. *S is a stable set.*

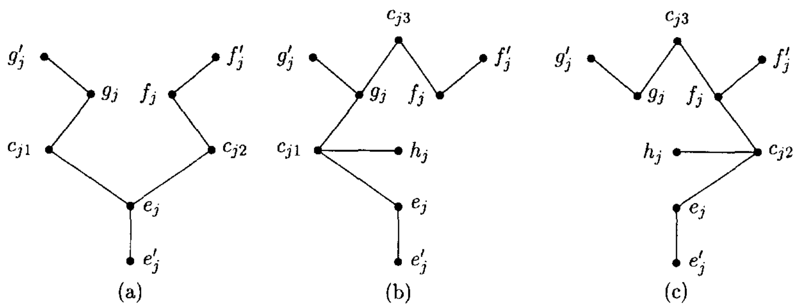
Proof. Since $c_{ij} \in S \Leftrightarrow b(c_j^{(i)}) = 0 \Leftrightarrow c_j^{(i)} \in T$ the steps (S1) and (S2) do not destroy the stability. If $b(c_1^{(j)}) = b(c_2^{(j)}) = 1$ then c_{j1}, $c_{j2} \notin S$ by (S2). Therefore h_j has no neighbors in S. □

Claim 8. *$G[v_i] - S$ is a tree $(i = 1, \ldots, n)$.*

Proof. Follows directly from the construction of S. □

Claim 9. *$G[C_j] - S$ is a tree $(j = 1, \ldots, m)$.*

Proof. Since in each clause exactly two literals are fulfilled there are only three possibilities for $G[C_j] - S$:



Note that by construction we have

$$c_j^{(i)} \in S \;\Leftrightarrow\; c_{ij} \notin S \quad (i = 1, \ldots, m,\; j = 1, 2, 3).$$

This together with Claims 8 and 9 show that $G - S$ is a tree. □

Proof of Theorem 2. We describe how one can change the construction in the proof of Lemma 1 such that G has maximum degree at most four. The only vertices which can have degree greater than four are v_i, $\overline{v_i}$ and r. Instead of $G[v_i]$ we now take $G'[v_i]$ as shown in Fig. 3.

We build G' from $G'[v_i]$, $i = 1, \ldots, n$, $G[C_j]$, $j = 1, \ldots, m$ and R by adding the following edges:

$$\begin{array}{ll}
v_j^{(i)} c_{jk} \text{ if } c_j^{(k)} = v_i & (i = 1, \ldots, n,\; j = 1, \ldots, m,\; k = 1, 2, 3), \\
\overline{v_j^{(i)}} c_{jk} \text{ if } c_j^{(k)} = \overline{v_i} & (i = 1, \ldots, n,\; j = 1, \ldots, m,\; k = 1, 2, 3), \\
a_i r,\; y_j^{(i)} r & (i = 1, \ldots, n,\; j = 1, \ldots, m), \\
e_j r & (j = 1, \ldots, m), \\
z_j^{(i)\prime} r & (i = 1, \ldots, n,\; j = 2, \ldots, m).
\end{array}$$

Then $\mathscr{C} \in$ 2-IN-3 3SAT holds if and only if G' has a good stable set. The proof is similar to the proof of Lemma 1, we only need the following additional claims:

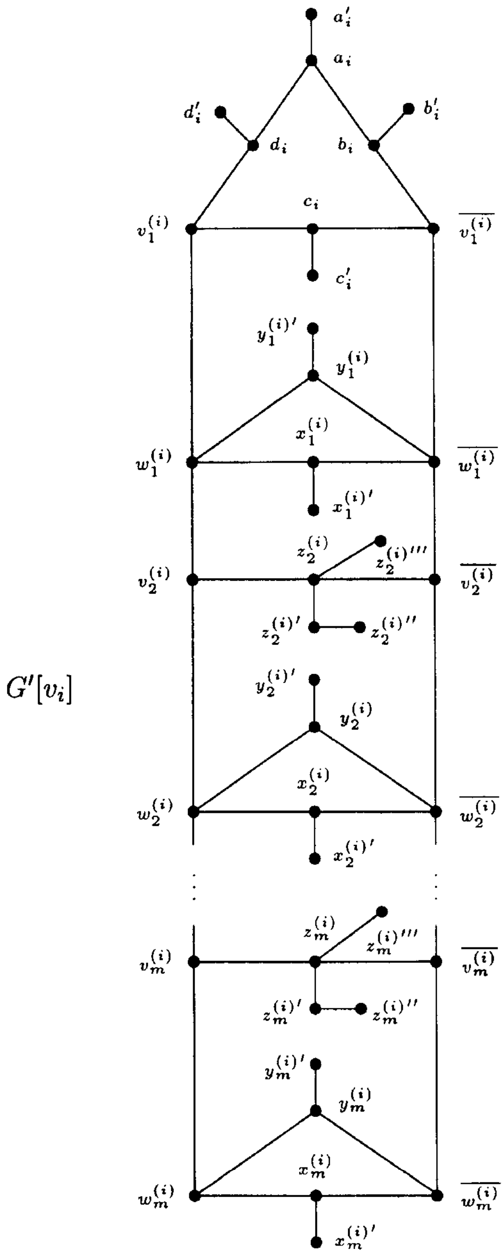


Fig. 3. The graph $G'[v_i]$.

For the “if” part we need the following claims.

Claim 10. *For every $i \in \{1,\dots,n\}$ the vertices of $\{v_j^{(i)}, \overline{w_j^{(i)}}\colon j=1,\dots,m\}$ resp. $\{\overline{v_j^{(i)}}, w_j^{(i)}\colon j=1,\dots,m\}$ all belong to S or T.*

Proof. W.l.o.g. let $v_1^{(i)} \in S$. By Claim 2 $\overline{v_1^{(i)}} \in T$. Since $y_1^{(i)}, x_1^{(i)} \in T$ and $G'(T)$ is cycle-free we get $w_1^{(i)} \in T$, $\overline{w_1^{(i)}} \in S$. Then $\overline{v_2^{(i)}} \in T$. Assume that $v_2^{(i)}$ is also in T. Since $x_2^{(i)} \in T$ not both $w_2^{(i)}$, $\overline{w_2^{(i)}}$ can be in S. W.l.o.g. let $w_2^{(i)} \in T$. Then $ry_2^{(i)}w_2^{(i)}v_2^{(i)}w_1^{(i)}y_1^{(i)}r$ induces a cycle in $G'(T)$, contradiction. The claim now follows by induction. □

Claim 11. *For every $i \in \{1,\dots,n\}$ the vertices of $G'[v_i]$ which are in T and r induce a tree in G'.*

Proof. Let

$$L_1 := \{a_i, a_i', b_i, b_i', c_i, c_i', d_i, d_i', v_1^{(i)}, \overline{v_1^{(i)}}\},$$
$$L_{2k} := \{w_k^{(i)}, \overline{w_k^{(i)}}, x_k^{(i)}, x_k^{(i)\prime}, y_k^{(i)}, y_k^{(i)\prime}\}, \quad k = 1,\dots,m,$$
$$L_{2k-1} := \{v_k^{(i)}, \overline{v_k^{(i)}}, z_k^{(i)}, z_k^{(i)\prime}, z_k^{(i)\prime\prime}, z_k^{(i)\prime\prime\prime}\}, \quad k = 2,\dots,m.$$

Then $L_1,\dots,L_{2m}$ is a partition of the vertices of $G'[v_i]$. By Claim 10 and since $v_1^{(i)} \in S \Leftrightarrow \overline{v_1^{(i)}} \in T$ for every k, $l \in \{1,\dots,2m\}$, $k \neq l$, there is no path from a vertex in $L_k \cap T$ to a vertex in $L_l \cap T$. Furthermore for every $k \in \{1,\dots,2m\}$ the vertices in $L_k \cap T$ induce a tree which settles the proof of this claim. □

Proof of Theorem 2 (*Conclusion*). For the "only if" part we must only change (S1) to (S1′) If $b(v_i) = 1$ put $v_j^{(i)}$, $\overline{w_j^{(i)}}$ $(j = 1,\dots,m)$ in S, otherwise put $\overline{v_j^{(i)}}$, $w_j^{(i)}$ $(j = 1,\dots,m)$.

G' is bipartite, a bipartition is given by

$$A := \{a_i', b_i, c_i, d_i, e_j', f_j', g_j', c_{j1}, c_{j2}, c_{j3}, w_j^{(i)}, \overline{w_j^{(i)}}, x_j^{(i)\prime}, y_j^{(i)\prime}, z_j^{(i)}, z_j^{(i)\prime\prime} : i = 1,\dots,n,$$
$$j = 1,\dots,m\} \cup \{r\},$$
$$B := \{a_i, b_i', c_i', d_i', e_j, f_j, g_j, h_j, v_j^{(i)}, \overline{v_j^{(i)}}, x_j^{(i)}, y_j^{(i)}, z_j^{(i)\prime}, z_j^{(i)\prime\prime\prime} : i = 1,\dots,n,$$
$$j = 1,\dots,m\} \cup \{r'\}.$$

Now, r is the only vertex in G' with degree greater than four. Let $u_1,\dots,u_l$ be the neighbors of r which are different from r'. We substitute R by R' shown in Fig. 4 and add u_ir_i, $i = 1,\dots,l$, as new edges.

It is not hard to prove that the resulting graph is bipartite with maximum degree four and that the reduction will also work with this graph. This completes the proof of Theorem 2. □

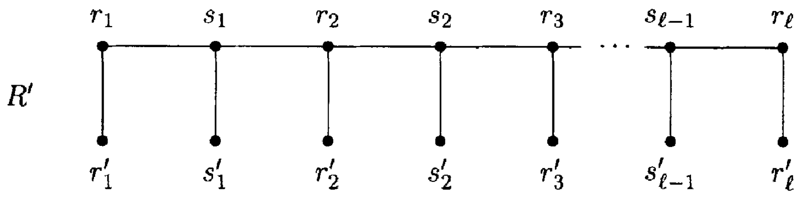


Fig. 4. The graph R'.

Remark. Clearly, Stable Tree can be solved for (bipartite) graphs with maximum degree at most 2. The complexity of Stable Tree for bipartite graphs with maximum degree at most three seems to be an open problem.

3. Proof of Theorem 3

The problem is clearly in NP. To prove the NP-completeness, we shall reduce 1-IN-3 3SAT to our problem. Let $\mathscr{C} = \{C_1, C_2, \ldots, C_m\}$ be any set of clauses $C_j = (c_{j1} \vee c_{j2} \vee c_{j3})$ given as input for 1-IN-3 3SAT, where the literals c_{jk} $(1 \leqslant j \leqslant m,\ 1 \leqslant k \leqslant 3)$ are taken from the set of variables V. We shall construct a graph G which has a good stable set if and only if $\mathscr{C}$ is satisfiable. For each variable $v \in V$ let $G(v,\bar{v})$ be the graph shown in Fig. 5. It is easy to see that

$G(v,\bar{v})$ has a good stable set. Every good stable set must contain exactly one of the labelled vertices v, $\bar{v}$.

For each clause $C_j = (c_{j1} \vee c_{j2} \vee c_{j3})$ let $G(C_j)$ be the graph shown in Fig. 5. It is easy to see that

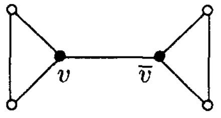
$G(C_j)$ has a good stable set. Every good stable set must contain exactly one of the labelled vertices c_{j1}, c_{j2}, c_{j3}.

Let G be the graph consisting of all graphs $G(v,\bar{v})$ and all $G(C_j)$, and edges xy between $G(v,\bar{v})$ and $G(C_j)$ if and only if $x \in \{v,\bar{v}\}$ is the literal $y \in \{c_{j1}, c_{j2}, c_{j3}\}$.

Now, assume that there is a truth assigment for $\mathscr{C}$. For each v, let $S(v,\bar{v})$ be a good stable set in $G(v,\bar{v})$ containing the false literal in $\{v,\bar{v}\}$. For each j, let S_j be a good stable set in $G(C_j)$ containing the true literal in $\{c_{j1}, c_{j2}, c_{j3}\}$. Set

$$S' = \bigcup_{v \in V} S(v,\bar{v}), \quad S'' = \bigcup_{j=1}^{m} S_j, \quad S = S' \cup S''.$$

Since there are no edges in G between the graphs $G(v,\bar{v})$ and no edges between the graphs $G(C_j)$, the sets S' and S'' are good stable sets in $\bigcup_{v \in V} G(v,\bar{v})$, respectively, in $\bigcup_{j=1}^{m} G(C_j)$. By construction of G and the definition of S' and S'', if xy is an edge of G with $x \in \{v,\bar{v}\}$ and $y \in \{c_{j1}, c_{j2}, c_{j3}\}$, then exactly one of x and y belongs to $S' \cup S''$. Thus, S is a stable set of G. Moreover, $G - S$ consists of $\bigcup_{v \in V} G(v,\bar{v}) - S'$ and $\bigcup_{j=1}^{m} G(C_j) - S''$, hence S is a good stable set of G.



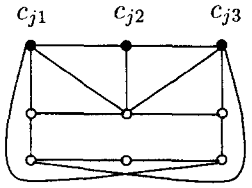


Fig. 5. Proof of Theorem 3, the graphs $G(v,\bar{v})$ (left) and $G(C_j)$ (right).

Conversely, suppose S is a good stable set of G. Then for each $v \in V$, exactly one of the labelled vertices v, $\bar{v}$ in G belongs to S. So the following assignment for $\mathscr{C}$

$$v = 1 \quad \text{if and only if} \quad v \notin S$$

is well-defined. Furthermore, for each j, S contains exactly one of the labelled vertices c_{j1}, c_{j2}, c_{j3} in $G(C_j)$, say y. The corresponding literal $x \in \{v, \bar{v}\}$ of y in $G(v, \bar{v})$ then cannot belong to S. So C_j has exactly one true literal. □

Remark. The same construction and arguments show that STABLE-"P_4-FREE" is NP-complete. Our graph G here, however, is not perfect (it contains chordless cycles of odd length at least five). In [11], a more complicated construction is discussed showing that STABLE-"P_4-FREE" is NP-complete for comparability graphs.

4. Proof of Theorems 4 and 5

Note that the graph G has a stable set S such that $G - S$ is partitionable into two cliques if and only if $\overline{G}$ has a stable set S' such that $\overline{G} - S'$ is partitionable into a stable set and a clique. Thus Theorem 4 is implied by Theorem 5 by considering complementation. So we only need to prove Theorem 5.

A *(k,l)-graph* is one whose vertex set can be partitioned into k stable sets and l cliques. Considering GRAPH 3-COLORABILITY, it is easy to see that deciding whether a graph is a (k,l)-graph is NP-complete for $k \geqslant 3$ or $l \geqslant 3$. Now Theorem 5 can be restated as follows: There is a polynomial-time algorithm to decide whether a graph is a (2,1)-graph. Such an algorithm was given in [1]. (The algorithm given in [2], however, is not correct; the necessity stated in [2, Proposition 1] does not hold.) We shall give here a more simple and more efficient algorithm than that in [1].

For convenience we denote by $(\boldsymbol{k}, \boldsymbol{l})$ the class of all (k,l)-graphs. For the vertex v, $N(v)$ (respectively, $\overline{N}(v)$) denotes the neighborhood (respectively, the non-neighborhood) of v. The basic principle of our approach for $(\mathbf{2},\mathbf{1})$ recognition is a vertex classification according to the following two neighborhood conditions

(N1) $N(v) \in (\mathbf{1},\mathbf{1})$;

(N2) $\overline{N}(v) \in (\mathbf{2},\mathbf{0})$.

Evidently, if a graph $G = (V, E)$ has a (2,1)-partition S_1, S_2, C with stable sets S_1, S_2 and clique C then for all vertices $v \in C$ condition (N2) is satisfied and for all vertices $v \in S_1 \cup S_2$ condition (N1) is satisfied. An immediate consequence is that if G has a vertex $v \in V$ with $N(v) \notin (\mathbf{1},\mathbf{1})$ and $\overline{N}(v) \notin (\mathbf{2},\mathbf{0})$ then $G \notin (\mathbf{2},\mathbf{1})$ holds.

Assume now that every vertex $v \in V$ satisfies at least one of the conditions (N1), (N2), and let G have a (2,1)-partition S_1, S_2, C. If $N(v) \notin (\mathbf{1},\mathbf{1})$ then necessarily v belongs to the clique C. If $\overline{N}(v) \notin (\mathbf{2},\mathbf{0})$ then necessarily v belongs to the bipartite part $S_1 \cup S_2$ of the (2,1)-partition.

Thus vertices satisfying exactly one of the conditions (N1), (N2) are assigned to their part of the (2,1)-partition (if there is such a partition). This means that the decision of "$G \in (\mathbf{2},\mathbf{1})$?" can be started by first assigning

$$C^F := \{v\colon\ N(v) \notin (\mathbf{1},\mathbf{1}) \text{ and } \overline{N}(v) \in (\mathbf{2},\mathbf{0})\},$$

$$B^F := \{v\colon\ N(v) \in (\mathbf{1},\mathbf{1}) \text{ and } \overline{N}(v) \notin (\mathbf{2},\mathbf{0})\}.$$

If C^F is no clique or B^F is not bipartite then evidently the correct answer is "$G \notin (\mathbf{2},\mathbf{1})$". Otherwise continue by considering the vertices satisfying both conditions (N1), (N2). Let

$$R := \{v\colon\ N(v) \in (\mathbf{1},\mathbf{1}) \text{ and } \overline{N}(v) \in (\mathbf{2},\mathbf{0})\}.$$

If $R = \emptyset$ then B^F and C^F describe a (2,1)-partition of G. Otherwise there is a vertex $x \in R$, and thus G has a (3,1)-partition defined by a (1,1)-partition S_1^x, C^x of $N[x]$ and a (2,0)-partition S_2^x, S_3^x of $\overline{N}(x)$. Subsequently we study whether this (3,1)-partition can be modified to a (2,1)-partition of G. This is done for the case $\omega(R) \geqslant 3$ by the following procedure.

Procedure Modify(x)

```
(P1) if $S_1^x \cup S_2^x \cup S_3^x$ is bipartite then $G \in (\mathbf{2},\mathbf{1})$. STOP
(P2) else for all $v \in S_1^x$ check whether $V \setminus (\{v\} \cup (C^x \cap N(v)))$ is bipartite;
(P3) if there is such a vertex $v$ then $G \in (\mathbf{2},\mathbf{1})$. STOP
```

Lemma 2. *If $\{a,b,c\}$ is a triangle in R then $G \in (\mathbf{2},\mathbf{1})$ if and only if for at least one vertex $x \in \{a,b,c\}$ Procedure* Modify(x) *leads to a (2,1)-partition of G.*

Proof. We only have to show that if $G \in (\mathbf{2},\mathbf{1})$ then the procedure successfully ends. Assume that S_1, S_2, C is a (2,1)-partition of G, and w.l.o.g. let C be a maximal clique. Then at least one of the vertices of a triangle a,b,c in R is in C, say $a \in C$. This implies that $C \subseteq N[a]$. Let S_1^a, C^a be a (1,1)-partition of $N[a]$ and let S_2^a, S_3^a be a (2,0)-partition of $\overline{N}(a)$. Since $C \subseteq N[a]$ and $|C \cap S_1^a| \leqslant 1$ and C is maximal we have either $C = C_a$ or $C = \{v\} \cup (C^a \cap N(v))$ for some $v \in S_1^a$. All these cases are checked by the procedure. □

Algorithm 1

```
Input:    A graph G.
Output:   A decision of "$G \in (\mathbf{2},\mathbf{1})$?".
(1)   for all $v \in V$ check whether $N(v) \in (\mathbf{1},\mathbf{1})$ and check whether $\overline{N}(v) \in (\mathbf{2},\mathbf{0})$;
(2)   if there is a vertex $v \in V$ with $N(v) \notin (\mathbf{1},\mathbf{1})$ and $\overline{N}(v) \notin (\mathbf{2},\mathbf{0})$ then
(3)      $G \notin (\mathbf{2},\mathbf{1})$. STOP
      else
(4)      $C^F := \{v\colon N(v) \notin (\mathbf{1},\mathbf{1})$ and $\overline{N}(v) \in (\mathbf{2},\mathbf{0})\}$;
(5)      $B^F := \{v\colon N(v) \in (\mathbf{1},\mathbf{1})$ and $\overline{N}(v) \notin (\mathbf{2},\mathbf{0})\}$;
(6)      if $C^F$ is no clique or $B^F$ is not bipartite then $G \notin (\mathbf{2},\mathbf{1})$. STOP
```

```
(7)       $R := \{v: N(v) \in (\mathbf{1},\mathbf{1})$ and $\overline{N}(v) \in (\mathbf{2},\mathbf{0})\}$;
(8)       if $R = \emptyset$ then $G \in (\mathbf{2},\mathbf{1})$. STOP
          else
(9)         if $\omega(R) \geqslant 3$ then choose a $K_3$ $\{a,b,c\} \subseteq R$ and for all $x \in \{a,b,c\}$ do
(10)          MODIFY($x$);
(11)          if for all $x \in \{a,b,c\}$ MODIFY($x$) unsuccessfully ends then
                $G \notin (\mathbf{2},\mathbf{1})$. STOP
              else $G \in (\mathbf{2},\mathbf{1})$. STOP
            else
(12)          check for all cliques $C^R \subseteq R$, $0 \leqslant |C^R| \leqslant 2$ whether
(13)          $C := C^F \cup C^R$ is a clique and $B := B^F \cup (R \setminus C^R)$ is bipartite;
(14)          if there is such a clique $C$ then $G \in (\mathbf{2},\mathbf{1})$. STOP
(15)          else $G \notin (\mathbf{2},\mathbf{1})$. STOP
```

Now we show that Theorem 5 holds:

Algorithm 1 *checks in* $\mathcal{O}(m^2)$ *steps whether a graph G is in* $(\mathbf{2},\mathbf{1})$.

Correctness. Up to step (10) the correctness of the algorithm is obvious. The correctness of the procedure follows from Lemma 2. The correctness of steps (13)–(15) is again obvious.

Time bound: It is well-known that the graphs in (1,1) and in (2,0) can be recognized in linear time $\mathcal{O}(n+m)$ (for the first class see [10]). Thus, step (1) can be done in time $\mathcal{O}(n(n+m))$. Obviously, steps (2)–(8) do not require more time. Step (9) can be done in time $\mathcal{O}(n(n+m))$ since a triangle is an edge where the two end-nodes have a common neighbor. Obviously, steps (10) and (11) can also be done in time $\mathcal{O}(n(n+m))$. In step (12), at most $n+m$ vertices and edges have to be considered. Step (13) takes at most $\mathcal{O}(n+m)$ time and has to be carried out for at most $n+m$ vertices and edges. Altogether, the time bound of the algorithm is $\mathcal{O}((n+m)^2)$. □

It is perhaps surprising that the case $\omega(R) \leqslant 2$ is the most time-consuming part of the algorithm. It could be done faster if there is an algorithm checking for a given graph G in less than $\mathcal{O}((n+m)^2)$ steps whether G has an edge e such that $G-e$ is bipartite.

The equivalence $G \in (\mathbf{1},\mathbf{2})$ if and only if $\overline{G} \in (\mathbf{2},\mathbf{1})$ implies Theorem 4: It can be recognized in $\mathcal{O}((n+\overline{m})^2)$ steps whether a graph G is in $(\mathbf{1},\mathbf{2})$.

Remark. It is easy to see that a similar approach leads to a polynomial-time algorithm for $(\mathbf{2},\mathbf{2})$ recognition using brute force in the case that there is a vertex $v \in V$ with $N(v) \in (\mathbf{1},\mathbf{2})$ and $\overline{N}(v) \in (\mathbf{2},\mathbf{1})$. In this case G has a (3,3)-partition and there is only a polynomial number of possible modifications of the (3,3)-partition to a (2,2)-partition. The details are described in [1].

5. Proof of Theorem 6

Recall that threshold graphs are split graphs. Thus, if $G \notin (\mathbf{2},\mathbf{1})$ then G has no partition into a stable set S and a threshold graph $G - S$. Assume now that $G \in (\mathbf{2},\mathbf{1})$. The fact that the intersection of a clique and a stable set contains at most one vertex implies that there are at most $\mathcal{O}(n^4)$ different partitions of G into a clique C and a bipartite graph $B = G(V \setminus C)$. According to Algorithm 1 and Theorem 5, all these partitions can be found in polynomial time. We consider now an arbitrary partition of this type: Let C be a clique of G such that $B = G(V \setminus C)$ is bipartite. We have to check whether B has a bipartition R,S such that $C \cup R$ or $C \cup S$ is a threshold graph. Now, let $Z_1,\dots,Z_k$ be the connected components of B. Note that every connected component Z_i, $i \in \{1,\dots,k\}$ of B has a unique bipartition R_i,S_i. Let $U_1,\dots,U_k$ be a choice such that $U_i \in \{R_i,S_i\}$, $i \in \{1,\dots,k\}$, and let $\overline{U_i}$ denote the other part in $\{R_i,S_i\}$, i.e. $\{U_i,\overline{U_i}\} = \{R_i,S_i\}$. It is easy to see that for every such choice $U_1 \cup \cdots \cup U_k$ and $\overline{U_1} \cup \cdots \cup \overline{U_k}$ define a bipartition of B, and every bipartition R,S of B has the property that $R = U_1 \cup \cdots \cup U_k$ and $S = \overline{U_1} \cup \cdots \cup \overline{U_k}$ for a suitable choice $U_1,\dots,U_k$.

Recall that a graph is a threshold graph if and only if it is $(P_4,C_4,2K_2)$-free, and a graph is a split graph if and only if it is $(C_5,C_4,2K_2)$-free (see [3]). Thus, every split graph is in particular $(C_4,2K_2)$-free. This implies that in order to check whether $C \cup R$ or $C \cup S$ induces a threshold graph for a clique C and stable sets R,S we have to check whether $C \cup R$ or $C \cup S$ induces a P_4-free graph.

For vertices $v \in V \setminus C$, let $N_C(v)$ denote the set of vertices of C adjacent to v. Vertices $u,v \in V \setminus C$ are called *C-incomparable* if $N_C(u)$ and $N_C(v)$ are incomparable w.r.t. set inclusion $\subseteq$. We call sets U_i,U_j, $i,j \in \{1,\dots,k\}$, $i \neq j$, *C-incomparable* if there are C-incomparable vertices $u \in U_i$ and $v \in U_j$. Note that C-incomparability of vertices and of sets U_i,U_j, $i,j \in \{1,\dots,k\}$, can be checked in polynomial time. The proof of the following claim is straightforward:

Claim 12. *For a bipartition R,S of B with $R = U_1 \cup \cdots \cup U_k$ and a clique C, $C \cup R$ is no threshold graph if and only if there is an index $i \in \{1,\dots,k\}$ such that $C \cup U_i$ is no threshold graph or there are $i,j \in \{1,\dots,k\}$, $i \neq j$, such that U_i and U_j are C-incomparable.*

We define now an instance F_B of the 2-Satisfiability Problem (2SAT) corresponding to B with bipartition $\{R_i,S_i\}$ of connected components Z_i, $i \in \{1,\dots,k\}$: Let $x_1,\dots,x_k$ be Boolean variables and F_B be the following conjunction consisting of one-literal and two-literal clauses. Hereby, for $U_i \in \{R_i,S_i\}$ let $l(U_i)$ be the literal which is x_i if $U_i = R_i$ and $\neg x_i$ if $U_i = S_i$.

(1) For every $U_i \in \{R_i,S_i\}$ such that $C \cup U_i$ is no threshold graph let $\neg l(U_i)$ be the corresponding clause.
(2) For all C-incomparable U_i,U_j with $U_i \in \{R_i,S_i\}$, $U_j \in \{R_j,S_j\}$ let $\neg(l(U_i) \wedge l(U_j))$ be the corresponding clause.

Claim 13. *F_B is satisfiable if and only if there is a bipartition R,S of B such that $C \cup R$ is a threshold graph.*

Proof of Claim 13. First assume that F_B is satisfied by a truth assignment b. Then define R as the union of all sets U_i, $i \in \{1,\dots,k\}$, for which $b(l(U_i))=1$. If in a clause with 2 literals both are satisfied, then take one of them. It is easy to see that this represents a choice of sets $U_i \in \{R_i, S_i\}$ such that $C \cup R$ induces a threshold graph.

Conversely, if there is a bipartition R,S of B such that $C \cup R$ is a threshold graph then let $b(x_i)=1$ if and only if $R_i \subseteq R$. Obviously, b defines a satisfying truth assignment of F_B. Thus the claim is proved. □

Summarizing, for every partition of G into a clique C and a bipartite B one has to consider the corresponding 2SAT problem which can be solved in linear time depending on the length of the input formula. If one of the instances F_B for bipartition B is satisfiable then G contains a stable set S such that $G-S$ is a threshold graph, otherwise not. The time bound $\mathcal{O}(n^8)$ can be seen as follows: For a fixed (2,1)-partition of G, one has to check the incomparabilities and to express them in an instance of 2SAT which then has to be checked for satisfiability. All this can be done in time $\mathcal{O}(n^4)$. Since there are at most $\mathcal{O}(n^4)$ different (2,1)-partitions of G which altogether can be found in time $\mathcal{O}(n^4)$ the total time bound is $\mathcal{O}(n^8)$. □

References

[1] A. Brandstädt, Partitions of graphs into one or two independent sets and cliques, in: Forschungsergebnisse der FSU Jena, N/84/71, 1984, Revised version: Informatik-Berichte FernUniversität Hagen No. 105, 1/1991, 1991.

[2] A. Brandstädt, Partitions of graphs into one or two independent sets and cliques, Discrete Math. 152 (1996) 47–54.

[3] A. Brandstädt, V.B. Le, J. Spinrad, Graph Classes – a Survey, SIAM Monographs in Discrete Math. Appl., SIAM, Philadelphia, 1998, to appear.

[4] L. Cai, D.G. Corneil, A generalization of perfect graphs – i-perfect graphs, J. Graph Theory 23 (1996) 87–103.

[5] M.R. Garey, D.S. Johnson, The complexity of near-optimal graph colouring, J. ACM 23(1) (1976) 43–49.

[6] M. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.

[7] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (1976) 237–267.

[8] M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (1981) 169–197.

[9] M. Grötschel, L. Lovász, A. Schrijver, Polynomial algorithms for perfect graphs, in: C. Berge, V. Chvátal (Eds.), Topics on Perfect Graphs, North-Holland, Amsterdam, 1984.

[10] P.L. Hammer, B. Simeone, The splittance of a graph, Combinatorica 1 (1981) 275–284.

[11] C.T. Hoàng, V.B. Le, On P_4-transversals in perfect graphs, Manuscript Fachbereich Informatik, Uni. Rostock, 1997, submitted.

[12] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–104.

[13] L. Lovász, Coverings and colorings of hypergraphs, 4th Southeastern Conf. Combinatorics, Graph Theory, and Computing, Utilitas Mathematica, 1973, pp. 3–12.
[14] F. Maffray, M. Preissmann, On the NP-completeness of the k-colorability problem for triangle-free graphs, Discrete Math. 162 (1996) 313–317.
[15] N.V.R. Mahadev, V.N. Peled, Threshold Graphs and Related Topics, Annals of Discrete Math., vol. 56, North-Holland, Amsterdam, 1995.
[16] B. Monien, personal correspondence, 1984.
[17] G. Sabidussi, Coloring problems for 4-regular Hamiltonian graphs, Kolloquium über Kombinatorik, TU Braunschweig, 14/15, November, 1997.
[18] T.J. Schaefer, The complexity of the satisfiability problem, Proc. 10th Ann. ACM Symp. on Theory of Computing, 1978, pp. 216–226.