Graphs and Combinatorics (2003) 19:347–356
Digital Object Identifier (DOI) 10.1007/s00373-002-0517-5

Graphs and Combinatorics

Robust Algorithms for the Stable Set Problem

Michael U. Gerber[1] and Vadim V. Lozin[2]

[1] Department of Mathematics, Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland. e-mail: michael.gerber@epfl.ch
[2] RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003, USA. e-mail: lozin@rutcor.rutgers.edu

Abstract. The stable set problem is to find in a simple graph a maximum subset of pairwise non-adjacent vertices. The problem is known to be NP-hard in general and can be solved in polynomial time on some special classes, like cographs or claw-free graphs. Usually, efficient algorithms assume membership of a given graph in a special class. Robust algorithms apply to any graph G and either solve the problem for G or find in it special forbidden configurations. In the present paper we describe several efficient robust algorithms, extending some known results.

Key words. Stable set, Stability number, Polynomial algorithm

1. Introduction

In [7], J. Spinrad proposed to call an algorithm solving a problem $\prod$ on a special graph class $\mathscr{C}$ *robust* if the algorithm on the input graph G either solves $\prod$ correctly or gives a witness for $G \notin \mathscr{C}$. Hence, before applying a robust algorithm to a graph G, there is no need to check if G is in $\mathscr{C}$. This can considerably reduce the computational effort if the time to recognize graphs in the class $\mathscr{C}$ is worse the time to solve the problem $\prod$. Furthermore, a robust algorithm might solve the problem correctly even if G is not in $\mathscr{C}$. In the present paper, we develop efficient robust algorithms for the stable set problem on several classes of graphs, extending some previously studied cases.

All graphs $G = (V, E)$ considered are undirected, without loops and multiple edges. By $N(v) = \{u : uv \in E\}$ we denote the *neighbourhood* of a vertex $v \in V$. The subgraph of G induced by a set of vertices U is denoted $G[U]$.

As usual, P_n (C_n) is the chordless *path* (*cycle*) on n vertices. We denote a path by $p_1p_2 \dots p_n$, and a cycle by $(c_1c_2 \dots c_n)$. The graph P_4 is of particular interest in this paper. In a $P_4 = abcd$, we call a and d the *endpoints*, and b and c the *midpoints* of the P_4. Graph $T_{i,j,k}$ is a tree with exactly three vertices of degree 1 being at distance i, j, k from the only vertex of degree 3. When $i = j = k = 1$, the graph is called a *claw*, and is denoted by $a\{b, c, d\}$, where a is the vertex of degree 3. When

$i = j = 1$ and $k = 2$, the graph is called a *chair* and is denoted by $\{a, b\}cde$, where c is the vertex of degree 3, and d the vertex of degree 2, and e is adjacent to d. A *banner* is the graph with vertices a, b, c, d, e and edges ab, ac, bd, cd and de. Such a banner is denoted by $a\{b, c\}de$. Some of the above graphs are shown in Fig. 1.

A *stable set* in a graph is a subset of vertices no two of which are adjacent. The maximum size of a stable set in a graph G is denoted $\alpha(G)$ and is called the *stability number* of G. The problem of finding a maximum stable set in a graph is known to be NP-hard in general. However, on special classes, like cographs [2], chordal [8] or claw-free graphs [5, 6] it can be solved in polynomial time. Efficient algorithms for the listed classes are not robust in the sense defined above. But it should be noticed that the recognition time for these classes coincides with the solution time. For such classes, the development of robust algorithms is not an actual task, and one can be satisfied by *pseudo-robust* ones consisting of two general steps: first, determining whether the input graph is in $\mathscr{C}$, and second, solving the problem. Consider, for example, cographs, i.e. graphs containing no induced path on four vertices P_4. Both recognizing cographs and solving the stable set problem for them can be realized in linear time [2]. This gives a natural pseudo-robust algorithm which, given a graph G, either solves the stable set problem for G or finds a P_4 in it. In the following, we shall refer to this algorithm as *Algorithm* $\mathscr{A}$.

In [1], the pseudo-robust algorithm for cographs became a base for the construction of a (strictly) robust one for the class of P_5 and banner-free graphs. The latter algorithm has been extended then to the class of P_5, $K_{3,3} - e$ and twin-house-free graphs. In Section 4 we consider another extension of P_5 and banner-free graphs defined by two forbidden induced subgraphs: $T_{2,2,2}$ and a banner. In addition to (P_5,banner)-free graphs, this class includes also all claw-free graphs that makes the problem much more difficult. The robust algorithm for ($T_{2,2,2}$,banner)-free graphs is based on the corresponding algorithm for (P_5,banner)-free graphs and uses, as an intermediate step, a pseudo-robust algorithm for the banner and chair-free graphs. The latter class also includes all claw-free graphs. A polynomial algorithm for the stable set problem in claw-free graphs is referred in the paper as *Algorithm* $\mathscr{B}$.

Another example of a robust algorithm can be found in [4]. It deals with (banner, $K_{2,3}$, C_5, $C_6, \ldots$)-free graphs which are an extension of chordal graphs. In Section 5, we generalize this result to (banner, C_5,$C_6, \ldots$)-free graphs including also all cographs. A diagram describing the inclusion relationship between classes under consideration is represented in Fig. 2.

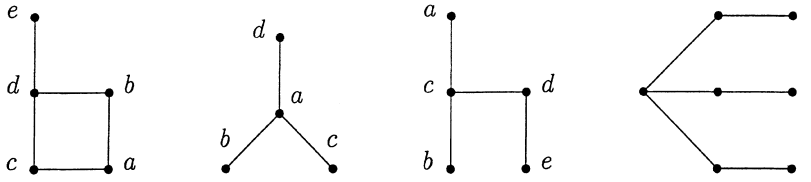

Fig. 1. Banner $a\{b, c\}de$, claw $a\{b, c, d\}$, chair $\{a, b\}cde$ and $T_{2,2,2}$

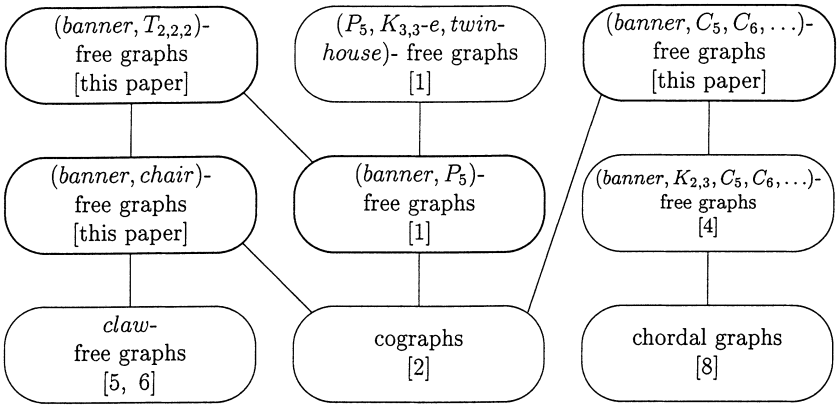

Fig. 2. Inclusion relationships between graph classes

2. Algorithm α

In order to make the paper self-contained, we describe in this section the robust algorithm presented in [1]. It either solves the stable set problem or finds an induced banner or an induced P_5 in a graph.

Algorithm α
Input: A graph G.
Output: A maximum stable set or a banner or a P_5 in G.

(1) Apply *Algorithm* $\mathscr{A}$ to G.
(2) If the output of Algorithm $\mathscr{A}$ is a stable set S, then let S be the output of Algorithm α and STOP; else let $abcd$ be a P_4 in G found by Algorithm $\mathscr{A}$.
(3) If there is a vertex x adjacent to a and c but not to b and d, then set the banner $a\{b,x\}cd$ as the output of the algorithm and STOP.
(4) If there are non-adjacent vertices x, x' adjacent to a and d but not to b, then set the banner $d\{x,x'\}ab$ as the output of the algorithm and STOP.
(5) If there is a vertex x adjacent to a but not to b, c and d, then set the P_5 $xabcd$ as the output of the algorithm and STOP.
(6) If there are non-adjacent vertices x and y such that x is adjacent to a and d but not to b, and y is adjacent to d but not to a and b, then set the P_5 $ydxab$ as the output of the algorithm and STOP.
(7) Let $G = G - b$ and go to (1).

To prove correctness of the algorithm we state the following lemma.

Lemma 1. *Let $abcd$ be an induced P_4 in the graph G and assume that $x, x' \in N(a) \backslash N(b)$ and $y \in N(d) \backslash N(b)$. If G contains no induced banner $a\{b,x\}cd$, $d\{x,x'\}ab$, and no induced P_5 $xabcd$, $ydxab$, then $\alpha(G - b) = \alpha(G)$.*

Proof. Let S be a stable set in G with $b \in S$, and let A denote the subset of vertices in S adjacent to a and different from b. We will show that G contains a stable set of the same size as S, but not containing b.

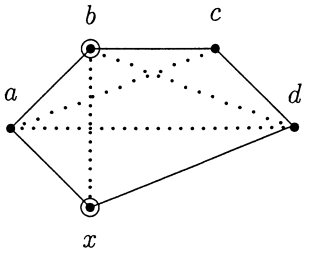

Fig. 3. Induced $P_4 = abcd$, with $b, x \in S$

If $A = \emptyset$, then obviously $S_1 := (S \backslash \{b\}) \cup \{a\}$ is a stable set in G. Assume now that $x \in A$. Then x must be adjacent to d, since otherwise G contains either a banner $a\{b,x\}cd$ (if x is adjacent to c), or a P_5 $xabcd$ (if x is not adjacent to c). This implies that d does not belong to S (see Fig. 3). We claim now that

(1) x is the only neighbor of a in A, and
(2) x is the only neighbor of d in S.

To prove (1), suppose that x' is another vertex in A. Then, similarly, x' is adjacent to d, but then $d\{x,x'\}ab$ is a banner in G. To show (2), assume that d has a neighbor $y \neq x$ in S. Since $ydxab$ is not permitted to induce a P_5 in G, it follows that $ya \in E$. But now a has a second neighbor in S, a contradiction to (1). From (1) and (2) it follows that $S_2 := (S \backslash \{b,x\}) \cup \{a,d\}$ is a stable set in G. □

Theorem 1. *Given a graph $G = (V,E)$, Algorithm α terminates in $O(|V||E|)$ steps, and if the output of the algorithm is a stable set S, then $|S| = \alpha(G)$.*

Proof. Algorithm $\mathscr{A}$ has time complexity $O(|V| + |E|)$ [2]. Verifying the conditions in steps (3) to (6) can be done in time $O(|E|)$. Algorithm α loops at most $|V|$ times through steps (1) to (7), hence its total time complexity is $O(|V||E|)$.

The proof of the second part of the theorem is a consequence of Lemma 1. □

3. Algorithm β

In this section we describe an algorithm which, for a given graph G, either finds an induced banner or an induced chair, or solves the stable set problem in G. This algorithm is pseudo-robust in the sense that it has a special stage for recognition of (banner,chair)-freeness (steps 1 and 2 of the algorithm). However, this stage does not have higher complexity than the solution stage (steps 3–9) in its present form. This gives us a reason to separate the recognition stage from the solution one in order to simplify the description of the algorithm.

Algorithm β
Input: A graph G.
Output: A maximum stable set or a banner or a chair in G.

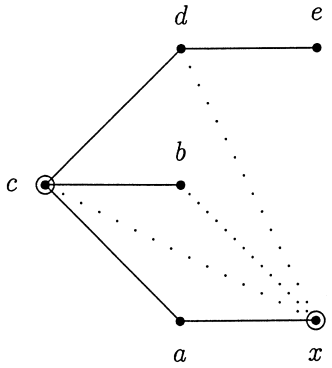

Fig. 4. Induced chair $\{a,b\}cde$, with $c,x \in S$

(1) If G contains an induced banner, then set the banner as the output of the algorithm and STOP.
(2) If G contains an induced chair, then set the chair as the output of the algorithm and STOP.
(3) Apply *Algorithm* α to G.
(4) If the output of Algorithm α is a stable set S, then let S be the output of Algorithm β and STOP; else let $abcde$ be a P_5 in G found by Algorithm α.
(5) Let A be the subset of vertices in G adjacent to every vertex of the found P_5, and B, the subset of vertices in G adjacent to every vertex in A (clearly $\{a,b,c,d,e\} \subseteq B$).
(6) In the subgraph $G[B]$ find the connected component H containing the P_5.
(7) Apply *Algorithm* $\mathscr{B}$ to graph H.
(8) Let S' be a maximum stable set in H found in step (7), and let $D = VH \backslash S'$, where VH is the set of vertices inducing H in G.
(9) Let $G = G - D$ and go to (3).

In the following, we state two lemmas and two corollaries which will help us to prove in Theorem 2 that if Algorithm β produces S as an output, then S is a maximum stable set. We start by proving the claim below, which will be applied in the proof of Lemma 2.

Claim 1. *Let $G = (V,E)$ be a (banner,chair)-free graph that contains an induced claw C. If a vertex $x \in V$ is not adjacent to the center of C, then x is either adjacent to no vertex in C, or to all vertices of degree* 1 *in C.*

Proof. Indeed, if x is adjacent to exactly one vertex in C, then vertices of C together with x induce a chair in G. And if x is adjacent to exactly two vertices in C, then C and x induce a banner. Hence the claim. □

Lemma 2. *Let $G = (V,E)$ be a connected (banner,chair)-free graph that contains an induced P_5. If in addition G contains an induced claw, then there is a vertex in G adjacent to every vertex of the P_5.*

Proof. The set of vertices of the P_5 will be denoted along the proof by $P = p_1p_2p_3p_4p_5$, and an induced claw will be denoted by $C = c_0\{c_1, c_2, c_3\}$, where c_0 is the center of the claw, i.e. the vertex of degree three.

From now on, we shall assume, without loss of generality, that C is a closest possible claw to P. Under this assumption C has a common vertex with P. To prove this, assume the contrary: let $x_1, \ldots, x_n$ be a shortest path connecting P to C in G, where $x_1 \in P$ and x_n is a vertex of C.

First we claim that x_{n-1} has at most two neighbors in set $\{c_1, c_2, c_3\}$, else $x_{n-1}\{c_1, c_2, c_3\}$ is an induced claw that is closer to P than C. Together with Claim 1 it implies that x_{n-1} is adjacent to c_0. Suppose now that x_{n-1} has two non-neighbors in $\{c_1, c_2, c_3\}$, say c_2 and c_3. Then $c_0\{x_{n-1}, c_2, c_3\}$ is an induced claw that is closer to P than C. Hence x_{n-1} has exactly two neighbors in set $\{c_1, c_2, c_3\}$, say c_1 and c_2. It follows that $x_{n-1} \in P$, otherwise $x_{n-1}\{c_1, c_2, x_{n-2}\}$ is an induced claw that is closer to P than C. Now let y be a neighbor of x_{n-1} along the P_5. We have yc_1 or yc_2 in E, otherwise $x_{n-1}\{y, c_1, c_2\}$ is an induced claw that has a common vertex with P. By symmetry, we can suppose that $yc_1 \in E$. Thus, (y, c_1) is another shortest path connecting P to C. Hence, like above, $yc_0 \in E$. In a similar way we obtain that every vertex of P is adjacent to c_0. But then $c_0\{p_1, p_3, p_5\}$ is an induced claw that is closer to P than C. This contradiction proves that C and P have a common vertex.

Without loss of generality we may assume that $c_0 \in P$. Indeed, if $c_0 \notin P$, then there must exist two adjacent vertices p_i and p_{i+1} of the path such that c_0 is adjacent to p_i but not to p_{i+1} (otherwise c_0 is adjacent to all vertices of the path proving the lemma). If p_{i+1} is adjacent to one of c_1, c_2 and c_3, then, by Claim 1, $p_{i+1}\{c_1, c_2, c_3\}$ is a claw with the center on the path. Suppose that p_{i+1} is adjacent to no vertex in C. In this case, if p_i has at least two non-neighbors in $\{c_1, c_2, c_3\}$, say c_1 and c_2, then the claw $c_0\{p_i, c_1, c_2\}$ and the vertex p_{i+1} contradict Claim 1. If p_i has at least two neighbors in $\{c_1, c_2, c_3\}$, say c_1 and c_2, then $p_i\{p_{i+1}, c_1, c_2\}$ is a claw with the center in P. Thus, from now on we have $c_0 \in P$. Up to the symmetry, we have to analyze the following cases: $c_0 = p_1$, $c_0 = p_2$ and $c_0 = p_3$.

Assume $c_0 = p_i$ with $i \in \{1, 2\}$, and let first $c_1 = p_{i+1}$. Then, by Claim 1, p_{i+2} is adjacent to c_2 and c_3, while p_{i+3} has no neighbors in $\{c_1, c_2, c_3\}$. But then $p_i\{c_2, c_3\}p_{i+2}p_{i+3}$ is a banner. Therefore, there is no claw centered at $c_0 = p_i$ with $i \in \{1, 2\}$ such that $c_1 = p_{i+1}$. Consequently, p_{i+1} has at least two neighbors in $\{c_1, c_2, c_3\}$, say c_2 and c_3. Now to avoid a fork $\{c_2, c_3\}p_{i+1}p_{i+2}p_{i+3}$ and a banner $p_i\{c_2, c_3\}p_{i+2}p_{i+3}$ and a banner $p_i\{c_2, c_3\}p_{i+3}p_{i+2}$ we must have that vertices p_{i+2} and p_{i+3} are adjacent both to c_2 and c_3 (by Claim 1). But then either c_2 or c_3 is adjacent to every vertex in P, otherwise $(P - \{p_{i+1}, p_{i+2}\}) \cup \{c_2, c_3\}$ induce a banner.

To complete the proof, we let $c_0 = p_3$ and assume there is no claw centered at p_i with $i \in \{1, 2\}$. Under this assumption, p_1 has no neighbors in $\{c_1, c_2, c_3\}$ (else $p_1\{c_1, c_2, c_3\}$ is a claw by Claim 1), and p_2 has at most one neighbor in $\{c_1, c_2, c_3\}$ (else $p_2\{p_1, c_1, c_2\}$ is a claw, assuming that c_1 and c_2 are adjacent to p_2). But now 2 non-neighbors of p_2 in $\{c_1, c_2, c_3\}$ together with p_1, p_2 and p_3 induce a fork. □

Corollary 1. *Let A be the subset of vertices in G adjacent to every vertex of the P_5, and B, the subset of vertices in G adjacent to every vertex in A, and H, the connected component of $G[B]$ containing the P_5, then H is a claw-free graph.*

Proof. Suppose that H is not claw-free. Then, by Lemma 3, there exists a vertex x in H that is adjacent to every vertex of the P_5, hence $x \in A$, a contradiction. □

Lemma 3. *Let $G = (V, E)$ be a connected (banner,chair)-free graph with an induced P_5, and A and B, the subsets of V as defined in Corollary 1, then there are no edges in G of form xy with $x \in B$ and $y \in V \backslash (A \cup B)$.*

Proof. Assume, to the contrary, vertices $x \in B$ and $y \in V \backslash (A \cup B)$ form an edge in the graph. Note that y has a non-neighbor z in A by definition of A and B.

Suppose first x is a vertex of the $P_5 = abcde$. If $x = b$, then y has a neighbor in $\{a, c\}$, otherwise G would contain either an induced banner $d\{y, c\}ba$ (if y is adjacent to d) or an induced chair $\{a, y\}bcd$ (if y is not adjacent to d). Similarly if $x = d$. Thus, in either case y has a neighbor in set $\{a, c, e\}$. It implies that y is adjacent to every vertex in $\{a, c, e\}$, otherwise vertices a, c, e, y, z would induce in G either a chair or a banner. Since $y \notin A$, it has a non-neighbor on the P_5, say b. But then G contains an induced banner $b\{a, c\}ye$, a contradiction.

Now assume y has no neighbors in the P_5, i.e. $x \notin \{a, b, c, d, e\}$. Clearly, x is not adjacent to every vertex of the P_5, otherwise x would belong to A. On the other hand, x must have a neighbor on the P_5, otherwise $\{a, c\}zxy$ is an induced chair in G. Hence, we can consider a pair of consecutive vertices of the P_5 one of which is adjacent to x, say p', and another one which is not adjacent to x, say p''. Moreover, there is a third vertex on the P_5 which is not adjacent to both vertices of the pair above, say p'''. For any possible choice of p', p'' and p''', we know that if x is adjacent to p''', then G contains an induced chair $\{p''', y\}xp'p''$, and if x is not adjacent to p''', then G contains an induced chair $\{p'', p'''\}zxy$. □

Corollary 2. *If H is the connected component of $G[B]$ containing the P_5, and S' is a maximum stable set in H, then $\alpha(G) = \alpha(G - (VH - S'))$, where VH is the set of vertices inducing H in G.*

Proof. Let R denote the set of vertices $V \backslash (A \cup B)$. Since every vertex in A is adjacent to every vertex in B, we have $\alpha(G) = \max(\alpha(G[A \cup R]), \alpha(G[B \cup R]))$. Furthermore, by Lemma 3, we have $\alpha(G) = \max(\alpha(G[A \cup R]), \alpha(G[B]) + \alpha(G[R]))$. Hence, it is possible to find a stable set in $G - (VH - S')$ that has the same size as a maximum stable set in G. □

Now, we can state the main result of this section:

Theorem 2. *Given a graph $G = (V, E)$, Algorithm β terminates in $O(|V|^5)$ steps, and if the output of the algorithm is a stable set S, then $|S| = \alpha(G)$.*

Proof. Determining if G contains an induced banner or an induced chair can be done in time complexity $O(|V|^5)$, since both graphs have 5 vertices. Algorithm β loops at most $|V|$ times through steps (3) to (9). Algorithm $\mathscr{B}$ can be implemented with time complexity $O(|V|^4)$ [3], and all other steps from (3) to (9) have lower time complexity. Hence, Algorithm β has total time complexity $O(|V|^5)$.

By Corollaries 1 and 2, we conclude that S is a maximum stable set. □

4. Algorithm γ

The algorithm presented in this section either solves the stable set problem in a graph G or finds an induced banner or an induced $T_{2,2,2}$.

Algorithm γ
Input: A graph G.
Output: A maximum stable set or a banner or a $T_{2,2,2}$ in G.

(1) Apply *Algorithm* β to G.
(2) If the output of Algorithm β is a stable set S, then let S be the output of Algorithm γ and STOP.
(3) If the output of Algorithm β is an induced banner, then let the banner be the output of Algorithm γ and STOP else let $\{a,b\}cde$ be an induced chair in G found by Algorithm β.
(4) If there is a $T_{2,2,2}$ in G induced by vertices a,b,c,d,x,y,z with $x \in N(a)\backslash(N(b) \cup N(c) \cup N(d))$, $y \in N(b)\backslash(N(a) \cup N(c) \cup N(d))$ and $z \in N(d)\backslash(N(a) \cup N(c) \cup N(b))$, then let the $T_{2,2,2}$ be the output of the algorithm and STOP.
(5) Let $G = G - c$ and go to (1).

We now state a lemma that will help us prove that Algorithm γ provides a maximum stable set S.

Lemma 4. *Let $\{a,b\}cde$ be an induced chair in a banner-free graph G. If G does not contain a $T_{2,2,2}$ induced by vertices a,b,c,d,x,y,z with $x \in N(a)\backslash(N(b) \cup N(c) \cup N(d))$ and $y \in N(b)\backslash(N(a) \cup N(c) \cup N(d))$ and $z \in N(d)\backslash(N(a) \cup N(c) \cup N(b))$, then $\alpha(G) = \alpha(G - c)$.*

Proof. Let S be a stable set in G containing vertex c. We shall show that there is a stable set S' in G such that S' does not contain c and $|S'| = |S|$.

If S does not contain vertices adjacent to a, except c, then $S' = (S\backslash\{c\}) \cup \{a\}$ is a required set. Assume now that x is a vertex in S adjacent to a ($x \neq c$). Then x is not adjacent to b. Indeed, if x is adjacent to b, then x is adjacent to d, otherwise G contains an induced banner $x\{a,b\}cd$, and hence x is adjacent to e else G contains an induced banner $a\{x,c\}de$, but then G contains an induced banner $c\{a,b\}xe$. As a consequence, x is not adjacent to d, otherwise G contains an induced banner $x\{a,d\}cb$, see Fig. 4.

If S does not contain vertices adjacent to b, except c, then $S' = (S\backslash\{c\}) \cup \{b\}$ is a required set. Suppose y is a vertex in S adjacent to b ($y \neq c$). Just as above, y is neither adjacent to a nor to d. We can state now that c is the only neighbor of d in S. Indeed, if z is another vertex in S adjacent to d, then z is adjacent neither to a nor to b by the arguments above. But then vertices a, b, c, d, x, y, z induce a $T_{2,2,2}$ that contradicts the assumption. Thus $S' = (S\backslash\{c\}) \cup \{d\}$ is a stable set as required. □

Theorem 3. *Given a graph $G = (V, E)$, Algorithm γ terminates in $O(|V|^6)$ steps, and if the output of the algorithm is a stable set S, then $|S| = \alpha(G)$.*

Proof. By Theorem 3, step (1) is of complexity $O(|V|^5)$. Step (4) takes at worst $O(|V|^3)$ time. And the loop (1)-(5) is carried out at most $|V|$ times. Hence the total time complexity of the algorithm is $O(|V|^6)$. By Lemma 4, we conclude that S is a maximum stable set. □

5. Algorithm δ

The algorithm of this section either solves the stable set problem in a graph G, or finds an induced banner or an induced cycle C_k of length $k \geq 5$. This graph class extends the class of $(banner, K_{2,3}, C_5, C_6, \ldots)$-free graphs studied by Mahadev [4].

Algorithm δ
Input: A graph $G = (V, E)$.
Output: A maximum stable set or a banner or a C_k ($k \geq 5$) in G.

(1) Apply *Algorithm* $\mathscr{A}$ to G.
(2) If the output of Algorithm $\mathscr{A}$ is a stable set S, then let S be the output of Algorithm δ and STOP; else let $abcd$ be an induced P_4 in G found by Algorithm $\mathscr{A}$.
(3) Extend $abcd$ to a maximal induced path $P_r = p_1 p_2 \ldots p_r$, i.e. find in G a maximal (with respect to inclusion) induced path P_r containing the $P_4 = abcd$.
(4) If there is a vertex x adjacent to p_1 and p_3 but not to p_2 and p_4, then set the banner $p_1\{p_2, x\}p_3p_4$ as the output of the algorithm and STOP.
(5) If there are non-adjacent vertices x, x' adjacent to p_1 and p_4 but not to p_2, then set the banner $p_4\{x, x'\}p_1p_2$ as the output of the algorithm and STOP.
(6) If there are non-adjacent vertices x and y such that $xp_1, xp_3, yp_3 \in E$ and $xp_2, yp_1, yp_2 \notin E$, then set the banner $p_1\{x, p_2\}p_3y$ as the output of the algorithm and STOP.
(7) If there is a vertex x adjacent to p_1 and p_k ($k \geq 4$), but not to $p_2, \ldots, p_{k-1}$ then set the cycle $(xp_1p_2 \ldots p_k)$ as the output of the algorithm and STOP.
(8) Let $G = G - p_2$ and go to (1).

Lemma 5. *Let $p_1p_2 \ldots p_r$ ($r \geq 4$) be a maximal inclusion-wise induced path P_r in the graph $G = (V, E)$ and assume that $x, x' \in N(p_1)\backslash N(p_2)$ and $y \in N(p_3)\backslash N(p_2)$. If G contains no induced banner $p_1\{p_2, x\}p_3p_4$, $p_4\{x, x'\}p_1p_2$, $p_1\{x, p_2\}p_3y$, and no induced cycle $(xp_1p_2 \ldots p_k)$ ($k \geq 4$), then $\alpha(G - p_2) = \alpha(G)$.*

Proof. Let S be a stable set in G with $p_2 \in S$, and let A denote the subset of vertices in S adjacent to p_1 and different from p_2. We will show that G contains a stable set of the same size as S, but not containing p_2.

If $A = \emptyset$, then obviously $S_1 := (S \backslash \{p_2\}) \cup \{p_1\}$ is a stable set in G.

Assume now that $x \in A$. By maximality of P_r, x has a neighbor in $\{p_3, \ldots, p_r\}$. Since G has no induced cycle $(xp_1p_2 \ldots p_k)$ with $k \geq 4$, x is adjacent to p_3. Furthermore, x is adjacent to p_4, otherwise G contains a banner $p_1\{p_2, x\}p_3p_4$. Obviously, p_3 does not belong to S. We claim now that

(1) x is the only neighbor of p_1 in A, and
(2) x is the only neighbor of p_3 in S.

To prove (1), suppose that x' is another vertex in A. Then, similarly, x' is adjacent to p_3 and p_4, but $p_4\{x, x'\}p_1p_2$ is a banner in G, a contradiction. To show (2) assume that p_3 has a neighbor $y \neq x$ in S. To avoid an induced banner $p_1\{x, p_2\}p_3y$, we have $yp_1 \in E$ that contradicts (1). From (1) and (2) it follows that $S_2 := (S \backslash \{p_2, x\}) \cup \{p_1, p_3\}$ is a stable set in G. □

Theorem 4. *Given a graph $G = (V, E)$, Algorithm δ terminates in $O(|V|^2|E|)$ steps, and if the output of the algorithm is a stable set S, then $|S| = \alpha(G)$.*

Proof. Algorithm $\mathscr{A}$ has time complexity $O(|V| + |E|)$. Step (3) can be executed in $O(|V||E|)$, while Steps (4)–(7) need $O(|V|^2)$. Since the algorithm loops at most $|V|$ times through steps (1)–(8), we have a total time complexity of $O(|V|^2|E|)$ for Algorithm δ.

The proof of the second part of this Theorem is a consequence of Lemma 5. □

References

1. Brandstädt, A., Lozin, V.V.: A note on α-redundant vertices in graphs. Discrete Appl. Math. **108**, 301–308 (2001)
2. Corneil, D.G., Perl, Y., Stewart, L.K.: A linear recognition algorithm for cographs. SIAM J. Comput. **14**, 926–934 (1985)
3. Lovász, L., Plummer, M.D.: Matching theory. Ann. Discrete Math. **29**, (1986)
4. Mahadev, N.V.R.: Vertex deletion and stability number, OR WR 90/2, Department of Mathematics, Swiss Federal Institute of Technology (1990)
5. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. J. Comb. Theory, Ser. B **28**, 284–304 (1980)
6. Sbihi, N.: Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. Discrete Math. **29**, 53–76 (1980)
7. Spinrad, J.P.: Representations of graphs, Book Manuscript, Vanderbilt University Nashville (TN) 1997
8. Gavril, F.: Algorithms for the minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM J. Comput. **1**, 180–187 (1972)

Received: October 2, 2001
Final version received: August 20, 2002