DIMACS Technical Report 2005-38
October 2005

A polynomial algorithm to find an independent set of maximum weight in a fork-free graph

by

Vadim V. Lozin [1]
RUTCOR, Rutgers University
640 Bartholomew Road, Piscataway
NJ 08854-8003, USA
E-mail: lozin@rutcor.rutgers.edu

Martin Milanič [2]
RUTCOR, Rutgers University
640 Bartholomew Road, Piscataway
NJ 08854-8003, USA
E-mail: mmilanic@rutcor.rutgers.edu

[1]Permanent Member
[2]Affiliated Graduate Student Member

ABSTRACT

The class of fork-free graphs is an extension of claw-free graphs and their subclass of line graphs. The first polynomial-time solution to the maximum weight independent set problem in the class of line graphs, which is equivalent to the maximum matching problem in general graphs, has been proposed by Edmonds in 1965 and then extended to the entire class of claw-free graphs by Minty in 1980. Recently, Alekseev proposed a solution for the larger class of fork-free graphs, but only for the unweighted version of the problem, i.e., finding an independent set of maximum cardinality. In the present paper, we describe the first polynomial-time algorithm to solve the problem for weighted fork-free graphs.

Keywords: Independent set; Polynomial-time algorithm

1 Introduction

An independent set in a graph G is a subset of vertices no two of which are adjacent. The MAXIMUM INDEPENDENT SET problem is that of finding in a graph an independent set of maximum cardinality. If each vertex of G is assigned a positive integer, the *weight* of the vertex, then we say that G is a *weighted graph*. The MAXIMUM WEIGHT INDEPENDENT SET problem consists in finding in a weighted graph an independent set of maximum total weight. This problem is known to be NP-hard in general. Moreover, it remains NP-hard even under substantial restrictions, for instance, for triangle-free graphs [17] and $K_{1,4}$-free graphs [15]. On the other hand, the problem can be solved in polynomial time for graphs in some special classes, such as line graphs—where it becomes equivalent to the problem of finding a maximum weight matching—or, more generally, claw-free graphs [15]. Recently, Alekseev [1] proposed a solution for the larger class of fork-free graphs, where a *fork* (also called a *chair*) is the graph obtained from a claw by a single subdivision of one of its edges. However, Alekseev's solution applies only to "unweighted" fork-free graphs. Besides, it has a high time complexity and uses a sophisticated approach which is difficult to implement. In the present paper, we propose the first polynomial-time algorithm to solve the problem for weighted fork-free graphs.

The organization of the paper is as follows. In the rest of this section we introduce basic notations and terminology. Section 2 provides historical remarks on the topic. Section 3 is devoted to the notion of modular decomposition of graph, which is one of the basic tools in our approach. Finally, Section 4 presents the solution.

All graphs in this paper are undirected, without loops and multiple edges. For a graph G, we denote by $V(G)$ the vertex set of G and by $E(G)$ its edge set. If G does not contain induced subgraphs isomorphic to a graph H, we say that G is H-free and call H a forbidden induced subgraph for G. As usual, P_n is a chordless path on n vertices, and $K_{n,m}$ is a complete bipartite graph with parts of size n and m. In particular, $K_{1,3}$ is a *claw*.

Given a graph G and a set $U \subseteq V(G)$, we denote by $G[U]$ the subgraph of G induced by U. If G is a weighted graph, then $\omega(U)$ is the weight of the set U, i.e. the sum of weights of its vertices, and $\alpha_\omega(G)$ is the weight of a maximum weight independent set in G. For a vertex $x \in V(G)$, we denote by $N(x)$ the neighborhood of x, i.e. the set of vertices adjacent to x. The complement of a graph G is denoted by $\overline{G}$ and is called co-G. Connected components of $\overline{G}$ will be called *co-components* of G. Given two disjoint subsets $U \subset V(G)$ and $W \subset V(G)$, we shall say that U *dominates* W if every vertex of U is adjacent to every vertex of W.

2 Historical remarks

In 1957, Berge proved that a matching in a graph is maximum if and only if there are no augmenting (alternating) chains with respect to the matching [2]. In 1965, Edmonds proposed the first polynomial-time algorithm to find an augmenting chain [9], thus solving the maximum matching problem, and then extended the solution to the weighted version of the problem [10]. Lovász and Plummer [13] observed that Edmonds' solution is "among the most involved of combinatorial algorithms".

By associating with a graph G its line graph $L(G)$ (i.e. $V(L(G)) = E(G)$ with two vertices being adjacent in $L(G)$ if and only if the respective edges of G have a vertex in common) one can transform the maximum matching problem for G into the maximum independent set problem for $L(G)$. Therefore, the maximum matching problem is equivalent to the maximum independent set problem restricted to the class of line graphs. It is known (and can be easily verified) that every line graph is claw-free. However, the claw is not the only minimal forbidden induced subgraph for line graphs (see e.g. [12] for the complete list of minimal non-line graphs). So, the class of line graphs constitutes a proper subclass of claw-free graphs. Interestingly enough, Berge's idea of augmenting chains can be extended from line graphs to the entire class of claw-free graphs, i.e. an independent set S in a claw-free graph is maximum if and only if there are no augmenting chains with respect to S. This fact motivated several researchers to study the independent set problem in the class of claw-free graphs. In 1980, Minty [15] and Sbihi [18] independently of each other found a solution to this problem. Minty's approach reduces the independent set problem in a claw-free graph to finding a maximum matching in an auxiliary graph, which he calls the "Edmonds' graph". Another reduction of the problem from claw-free to line graphs (and hence to finding a maximum matching) exploiting an entirely different approach was proposed by Lovász and Plummer in [13]. The Lovász-Plummer solution, as well as Sbihi's, applies to "unweighted" graphs only. Lovász and Plummer remarked that "with some care, the algorithm can be implemented in $O(n^4)$ time", while Sbihi claimed an $O(n^3)$ solution. Minty did not give any time bound on his algorithm, but he claimed a solution to the problem for the case of weighted graphs. However, Nakamura and Tamura [16] recently found that the weighted version of Minty's algorithm fails for some special cases and proposed modifications to overcome this defect. Unfortunately, they also avoided giving any time bound on the solution and restricted themselves to claiming polynomial-time solvability of the problem in the case of weighted claw-free graphs. A concise but thorough description of Minty's algorithm and its extension to the weighted case can be found in [19].

For a long time the class of claw-free graphs remained one of the only three maximal graph classes defined by a single forbidden induced subgraph where the maximum weight independent set problem was known to be solvable in polynomial time, the other two being P_4-free graphs [7] and mK_2-free graphs [11]. Recently, Alekseev [1] found a polynomial-time solution for fork-free graphs, extending both claw-free and P_4-free graphs, but only for the unweighted version of the problem. He generalized the idea of augmenting chains by showing that in the class of fork-free graphs there are two types of augmenting graphs: chains and complexes, i.e. bipartite graphs whose every vertex contains at most one non-neighbor in the opposite part. To find an augmenting complex, Alekseev proposed a sophisticated construction that recursively reduces the problem to claw-free graphs. Purposely, he also avoids estimating the time complexity of his algorithm.

In the present paper we develop a completely different approach to the problem, which allows us to extend polynomial-time solvability to the case of weighted fork-free graphs. Along with claw-free and P_4-free graphs, our result generalizes several other solutions on subclasses of fork-free graphs, such as (fork,bull)-free [8, 3] and some other classes [5, 6].

3 Modular decomposition

Let $G = (V, E)$ be a graph, U a subset of V and x a vertex of G outside U. We shall say that x *distinguishes* U if x has both a neighbor and a non-neighbor in U. A subset $U \subset V(G)$ is called a *module* in G if it is indistinguishable for the vertices outside U. A module U is *nontrivial* if $1 < |U| < |V|$, otherwise it is *trivial*. A graph whose every module is trivial is called *prime*.

An important property of maximal modules is that if G and co-G are both connected, then the maximal modules of G are pairwise disjoint. Moreover, from the above definition it follows that if U and W are maximal modules, then either U dominates W or there are no edges between them. This property provides a reduction of the maximum weight independent set problem (and many other problems) from the graph G to a graph G^0 obtained from G by contracting each maximal module to a single vertex. We formally describe this reduction in the recursive procedure ALPHA(G) below.

Algorithm ALPHA(G)

Input: a weighted graph G
Output: an independent set S of maximum weight in G.

1. If $|V(G)| = 1$, set $S = V(G)$ and go to 7.
2. If G is disconnected, partition it into connected components $M_1, \dots, M_k$.
3. If co-G is disconnected, partition G into co-components $M_1, \dots, M_k$.
4. If G and co-G are connected, partition G into maximal modules $M_1, \dots, M_k$.
5. Construct a weighted graph G^0 from G by contracting each M_j ($j = 1, \dots, k$) to a single vertex and assigning to that vertex the weight $w(\text{ALPHA}(G[M_j]))$.
6. Find in G^0 an independent set S^0 of maximum weight, and set $S = \bigcup_{j \in S^0} \text{ALPHA}(G[M_j])$.
7. Return S and STOP.

Observe that the graph G^0 constructed in step 5 of the algorithm is either an edgeless graph, a complete graph, or a prime graph. Therefore, the modular decomposition approach reduces the problem from a graph to its prime induced subgraphs. The following theorem answers the question on the complexity of such a reduction.

Theorem 1 *Let X be a class of graphs and X^* the class of all prime induced subgraphs of the graphs in X. If the maximum weight independent set problem can be solved for graphs in X^* in time $O(n^p)$ for some constant $p \geq 2$, then this problem can be solved for graphs in X also in time $O(n^p)$.*

Proof. Let G be a graph in X with n vertices and m edges. The recursive decomposition of G produced by *Algorithm* ALPHA can be implemented in time $O(n+m)$ [14]. This decomposition associates with G a tree $T(G)$ whose leaves correspond to the vertices of G, while the internal nodes of $T(G)$ represent induced subgraphs of G with at least two vertices.

Consider an internal node U of $T(G)$, and let G_U denote the induced subgraph of G corresponding to U. Then the children of G_U correspond to the subgraphs $G[M_1], \dots, G[M_k]$, where $\{M_1, \dots, M_k\}$ is the partition of G_U defined in steps 2–4 of the algorithm. If G_U ($\overline{G}_U$) is disconnected, then G^0_U is an empty (complete) graph, and the problem can be trivially solved for G^0_U in time $O(|V(G^0_U)|)$. If both G and $\overline{G}$ are connected, then G^0_U is a prime induced subgraph of G, and the problem can be solved for G^0_U in time $O(|V(G^0_U)|^p)$ with $p \geq 2$ by our assumption. Summing up over all internal nodes of $T(G)$, we conclude that the total time complexity of the problem on G is bounded by $O(\sum\limits_U |V(G^0_U)|^p)$. It is not difficult to see that the total number of vertices in all graphs G^0_U corresponding to internal nodes $U \in V(T(G))$ equals to the number of edges of $T(G)$, i.e. $|V(T(G))| - 1$. Since the number of leaves of $T(G)$ is n and the number of internal nodes is at most $n-1$, we conclude that

$$\sum_U |V(G^0_U)|^p \leq (\sum_U |V(G^0_U)|)^p \leq (2n-2)^p = O(n^p).$$

The theorem is proved. ■

4 The solution

For a graph G and a vertex $x \in V(G)$, let us denote by G_x the graph obtained from G by deleting x and every vertex adjacent to x. Our solution to the maximum weight independent set problem for fork-free graphs is based on two general tools: modular decomposition and the following obvious identity

$$\alpha_\omega(G) = \max_{x \in V(G)} \{\omega(x) + \alpha_\omega(G_x)\}.$$

An immediate consequence of this identity is the following proposition.

Proposition 1 *If for every vertex $x \in V(G)$, the maximum weight independent set problem can be solved for G_x in time T, then it can be solved for G in time nT.*

Now we prove the main result that leads to an efficient solution of the problem in the class of fork-free graphs.

Theorem 2 *Let G be a fork-free graph, x an arbitrary vertex of G, and $\tilde{G}$ an induced subgraph of G_x. If both G and $\tilde{G}$ are prime, then $\tilde{G}$ is claw-free.*

Proof. Assume by contradiction that $\tilde{G}$ contains an induced claw. Then it must contain one of the minimal prime extensions of the claw. The complete list of such extensions can be found in [4]. It consists of 12 graphs, 7 of which contain a fork, and the remaining 5 are represented in Figure 1.

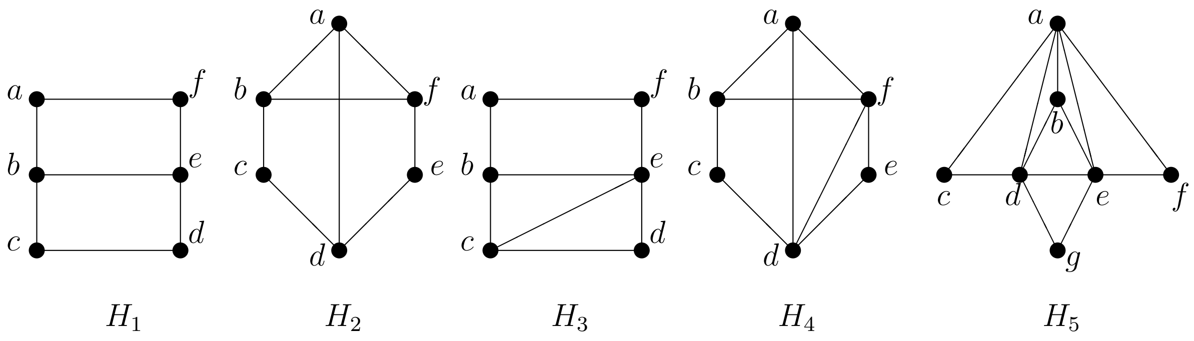

Figure 1. Minimal fork-free prime extensions of the claw

Let $H \in \{H_1, \ldots, H_5\}$ denote an induced copy of one of the minimal fork-free extensions of the claw in the graph $\tilde{G}$.

Claim 1 *No neighbor of x distinguishes $V(H)$.*

The proof of this claim is a rather tedious case analysis. For the purpose of readability of our current proof, we postpone it for the next section. The statement of the claim allows us to partition the set of neighbors of x into two sets Y and Z such that no vertex in Y has a neighbor in $V(H)$, while Z dominates $V(H)$.

Let W be an (inclusionwise) maximal subset of vertices of G_x with the following properties:

(i) $W \supseteq V(H)$,

(ii) $G[W]$ is connected,

(iii) co-$G[W]$ is connected,

(iv) Z dominates W,

(v) there are no edges between W and Y.

Note that such a set W exists since $V(H)$ satisfies all these properties. By definition, $|W| < |V(G)|$. In addition, $|W| \geq |V(H)| > 1$. Therefore, in order to be prime, G must contain a vertex $u \in V(G) \backslash W$ that distinguishes W. We will now show that the set $W' := W \cup \{u\}$ also enjoys the five stated properties.

Firstly, properties (iv) and (v) for W imply that $u \in V(G_x)$, which yields $W' \subseteq V(G_x)$. W' trivially satisfies property (i).

Since W enjoys properties (ii) and (iii) and since u has both a neighbor and a non-neighbor in W, we conclude that the same two properties still hold for W'.

To see that W' satisfies (iv), i.e. that Z dominates W', assume to the contrary that u has a non-neighbor z in Z. Since u distinguishes W and the complement of $G[W]$ is connected, u distinguishes a pair of non-adjacent vertices $w_1, w_2 \in W$. But now, a fork arises on $\{u, w_1, z, w_2, x\}$, contradicting the fork-freeness of G.

Finally, let us show that there are no edges between W' and Y. Indeed, suppose u is adjacent to a vertex $y \in Y$. Let $P = (v_0, \ldots, v_k)$ be a shortest path connecting $V(H)$ to u in the graph $G[W']$, i.e. $v_0 \in V(H)$ and $v_k = u$. Also, denote $v_{k+1} := y$, $v_{k+2} := x$. Since v_2 has no neighbors in $V(H)$, we conclude by analogy with Claim 1 that v_1 dominates $V(H)$. But now any two non-adjacent vertices of $V(H)$ together with v_1, v_2 and v_3 induce a fork.

Therefore, the subset W' of $V(G_x)$ satisfies all the above properties, thus contradicting the maximality of W. This completes the proof of Theorem 2. ■

Combining Theorems 1 and 2 with Proposition 1, we conclude that

Theorem 3 *The maximum weight independent set problem in the class of fork-free graphs can be solved in polynomial time. In particular, it can be solved in time nT, where T is the time to solve the same problem for claw-free graphs.*

Proof. Let X be the class of fork-free graphs, and X^* the class of prime graphs in X. Also, define $Y := \{G_v \ : \ G \in X^* \text{ and } v \in V(G)\}$ and let Y^* denote the class of all prime induced subgraphs of the graphs in Y. By Theorem 2, every graph in Y^* is claw-free. According to [15, 16], the maximum weight independent set problem in the class of claw-free graphs can be solved in polynomial time T. Therefore, by Theorem 1, the problem can be solved for graphs in Y also in time T. This implies an nT solution for graphs in X^* (by Proposition 1) and an nT solution for graphs in X (again by Theorem 1). ■

5 Proof of Claim 1

Let y be a neighbor of x. We present the proof for each of the cases $H = H_i$ for $i \in \{1, \ldots, 5\}$. We denote the vertices of H as depicted in Figure 1.

I. $H = H_1$.

Case 1: y has a non-neighbor among the vertices of degree 2 in H. Taking into account the symmetry, we may assume without loss of generality that y is not adjacent to a. We consider two subcases.

1.1: y is adjacent to c. Then

- y is not adjacent to f (otherwise a fork arises on $\{a, f, y, c, x\}$),
- y is not adjacent to e (otherwise a fork arises on $\{f, e, y, c, x\}$),
- y is not adjacent to b (otherwise a fork arises on $\{x, y, b, a, e\}$),
- y is not adjacent to d (otherwise a fork arises on $\{y, d, e, b, f\}$).

But now a fork arises on $\{x, y, c, b, d\}$, a contradiction.

1.2: y is not adjacent to c. Then

- y is not adjacent to b (otherwise a fork arises on $\{x, y, b, a, c\}$),
- y is not adjacent to e (otherwise a fork arises on $\{y, e, b, a, c\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, c, e\}$), and, by symmetry, to f.

Case 2: y is adjacent to every vertex of degree 2 in H. Then y is adjacent to b, since otherwise a fork would arise on $\{x, y, a, d, b\}$. By symmetry, y is adjacent to e.

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 2) or to none of them (Case 1.2), and the claim for the case $H = H_1$ follows.

II. $H = H_2$.

Case 1: y is adjacent to a vertex of degree 2 in H. Taking into account the symmetry, we may assume without loss of generality that y is adjacent to c. We consider two subcases.

1.1: y is adjacent to f. Then

- y is adjacent to d (otherwise a fork arises on $\{d, c, y, f, x\}$,
- y is adjacent to e (otherwise a fork arises on $\{e, f, y, c, x\}$,
- y is adjacent to b (otherwise a fork arises on $\{b, c, y, e, x\}$),
- y is adjacent to a (otherwise a fork arises on $\{a, b, y, e, x\}$).

1.2: y is not adjacent to f. Then

- y is not adjacent to a (otherwise a fork arises on $\{f, a, y, c, x\}$),
- y is not adjacent to e (otherwise a fork arises on $\{f, e, y, c, x\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, a, e\}$).

But now, a fork arises on $\{y, c, d, a, e\}$, a contradiction.

Case 2. y has no neighbor among the vertices of degree 2 in H. Then

- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, c, e\}$),
- y is not adjacent to a (otherwise a fork arises on $\{y, a, d, c, e\}$),
- y is not adjacent to b (otherwise a fork arises on $\{x, y, b, a, c\}$), and, by symmetry, to f.

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 1.1) or to none of them (Case 2), and the claim for the case $H = H_2$ follows.

III. $H = H_3$.

Case 1: y is adjacent to d. We consider two subcases.

1.1: y is adjacent to f. Then

- y is adjacent to a (otherwise a fork arises on $\{a, f, y, d, x\}$),
- y is adjacent to c (otherwise a fork arises on $\{c, d, y, f, x\}$),
- y is adjacent to b (otherwise a fork arises on $\{b, c, y, f, x\}$),
- y is adjacent to e (otherwise a fork arises on $\{e, c, y, a, x\}$).

1.2: y is not adjacent to f. Then

- y is not adjacent to a (otherwise a fork arises on $\{f, a, y, d, x\}$),
- y is not adjacent to b (otherwise a fork arises on $\{a, b, y, d, x\}$),
- y is not adjacent to e (otherwise a fork arises on $\{x, y, e, b, f\}$).

But now, a fork arises on $\{y, d, e, b, f\}$, a contradiction.

Case 2. y is not adjacent to d. We consider two subcases.

2.1: y is adjacent to f. Then

- y is not adjacent to c (otherwise a fork arises on $\{d, c, y, f, x\}$),
- y is not adjacent to b (otherwise a fork arises on $\{c, b, y, f, x\}$).
- y is not adjacent to e (otherwise a fork arises on $\{x, y, e, b, d\}$).

But now, a fork arises on $\{y, f, e, b, d\}$, a contradiction.

2.2: y is not adjacent to f. Then

- y is not adjacent to e (otherwise a fork arises on $\{x, y, e, d, f\}$),
- y is not adjacent to b (otherwise a fork arises on $\{y, b, e, d, f\}$),
- y is not adjacent to a (otherwise a fork arises on $\{x, y, a, b, f\}$),
- y is not adjacent to c (otherwise a fork arises on $\{x, y, c, b, d\}$).

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 1.1) or to none of them (Case 2.2), and the claim for the case $H = H_3$ follows.

IV. $H = H_4$.

Case 1. y is adjacent to c. We consider two subcases.

1.1: y is adjacent to e. Then

- y is adjacent to b (otherwise a fork arises on $\{b, c, y, e, x\}$),
- y is adjacent to f (otherwise a fork arises on $\{f, e, y, c, x\}$),
- y is adjacent to a (otherwise a fork arises on $\{a, b, y, e, x\}$),
- y is adjacent to d (otherwise a fork arises on $\{d, e, y, b, x\}$).

1.2: y is not adjacent to e. Then

- y is not adjacent to f (otherwise a fork arises on $\{e, f, y, c, x\}$),
- y is not adjacent to a (otherwise a fork arises on $\{f, a, y, c, x\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, a, e\}$).

But now, a fork arises on $\{y, c, d, a, e\}$, a contradiction.

Case 2: y is not adjacent to c. We consider two subcases.

2.1: y is adjacent to e. Then

- y is not adjacent to b (otherwise a fork arises on $\{c, b, y, e, x\}$),
- y is not adjacent to a (otherwise a fork arises on $\{b, a, y, e, x\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, a, c\}$).

But now, a fork arises on $\{y, e, d, a, c\}$, a contradiction.

2.2: y is not adjacent to e. Then

- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, c, e\}$),
- y is not adjacent to a (otherwise a fork arises on $\{y, a, d, c, e\}$),
- y is not adjacent to b (otherwise a fork arises on $\{x, y, b, a, c\}$),
- y is not adjacent to f (otherwise a fork arises on $\{x, y, f, a, e\}$).

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 1.1) or to none of them (Case 2.2), and the claim for the case $H = H_4$ follows.

V. $H = H_5$.

Case 1: y is adjacent to c. We consider two subcases.

1.1: y is adjacent to f. Then

- y is adjacent to d (otherwise a fork arises on $\{d, c, y, f, x\}$), and, by symmetry, to e,
- y is adjacent to b (otherwise a fork arises on $\{b, d, y, f, x\}$),
- y is adjacent to g (otherwise a fork arises on $\{g, d, y, f, x\}$),
- y is adjacent to a (otherwise a fork arises on $\{a, b, y, g, x\}$).

1.2: y is not adjacent to f. Then

- y is not adjacent to e (otherwise a fork arises on $\{f, e, y, c, x\}$),
- y is not adjacent to b (otherwise a fork arises on $\{e, b, y, c, x\}$),
- y is not adjacent to g (otherwise a fork arises on $\{e, g, y, c, x\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, b, g\}$).

But now, a fork arises on $\{y, c, d, b, g\}$, a contradiction.

Case 2. y is not adjacent to c. We may assume that y is not adjacent to f, since otherwise we are in a case symmetric to the case *1.2.* Then

- y is not adjacent to a (otherwise a fork arises on $\{x, y, a, c, f\}$),
- y is not adjacent to b (otherwise a fork arises on $\{y, b, a, c, f\}$),
- y is not adjacent to d (otherwise a fork arises on $\{x, y, d, b, c\}$), and, by symmetry, to e,
- y is not adjacent to g (otherwise a fork arises on $\{y, g, d, b, c\}$).

Therefore, every neighbor of x is adjacent either to all vertices of H (Case 1.1) or to none of them (Case 2), and the claim for the case $H = H_5$ follows. This completes the proof of the claim.

6 Conclusion

This paper proposes a polynomial-time solution to the maximum weight independent set problem in the class of fork-free graphs by reducing it to claw-free graphs. A solution for claw-free graphs is based on a reduction to line graphs, where the problem is equivalent to that of finding a maximum weight matching in a general graph. The main approach to the maximum matching problem exploits the idea of augmenting chains proposed by Berge and implemented by Edmonds. The same idea allows one to extend the solution from line graphs to claw-free graphs. In particular, in the case of unweighted claw-free graphs, the problem can be solved in $O(n^3)$ time, due to a result of Sbihi [18]. Minty's algorithm that solves the problem for weighted claw-free graphs (with a correction proposed by Nakamura and Tamura) requires $O(n^7)$ time. A further extension of the idea of augmenting graphs to the class of fork-free graphs also leads to a polynomial solution, but only in the unweighted case. Besides, the time complexity increases drastically. A rough analysis of Alekseev's solution suggests a time complexity of $O(n^{10})$.

To extend the solution to weighted fork-free graphs, we used an entirely different approach, and the extra time required for this extension is not critical. To improve the overall time complexity, one needs a better reduction from weighted claw-free to weighted line graphs, which seems to be a challenging research problem. Most of the forbidden graphs characterizing the class of line graphs are not prime, and many of them contain a clique separator. This suggests the idea of combining the modular decomposition technique with finding clique separators to reduce the problem from claw-free to line graphs. If such a reduction is possible, it would probably improve the time complexity greatly. We leave this question as an open problem for future research.

References

[1] V.E. Alekseev, A polynomial algorithm for finding the largest independent sets in fork-free graphs, *Discrete Applied Mathematics*, 135 (2004) 3–16.

[2] C. Berge, Two theorems in graph theory, *Proc. Nat. Acad. Sci. USA*, 43 (1957) 842–844.

[3] A. Brandstädt, T.C. Hoáng and V.B. Le, Stability number of bull- and chair-free graphs revisited, *Discrete Applied Mathematics*, 131 (2003) 39–50.

[4] A. Brandstädt, T.C. Hoáng and J.-M. Vanherpe, On minimal prime extensions of a four-vertex graph in a prime graph, *Discrete Mathematics*, 288 (2004) 9–17.

[5] A. Brandstädt, V.B. Le and N.H. de Ridder, Efficient robust algorithms for the maximum weight stable set problem in chair-free graph classes, *Information Processing Letters*, 89 (2004) 165–173.

[6] A. Brandstädt, H.-O. Le and J.-M. Vanherpe, Structure and stability number of chair-, co-P- and gem-free graphs revisited, *Information Processing Letters* 86 (2003) 161–167.

[7] D.G. Corneil, H. Lerchs and L. Stewart-Burlingham, Complement reducible graphs, *Discrete Applied Mathematics*, 3 (1981) 163–174.

[8] C. De Simone and A. Sassano, Stability number of bull- and chair-free graphs, *Discrete Applied Mathematics*, 41 (1993) 121–129.

[9] J. Edmonds, Path, trees, and flowers, *Canadian J. Mathematics*, 17 (1965) 449–467.

[10] J. Edmonds, Maximum matching and a polyhedron with $0,1$-vertices, *J. Res. Nat. Bur. Standards Sect. B* 69B (1965) 125–130.

[11] M. Farber, M. Hujter and Zs. Tuza, An upper bound on the number of cliques in a graph, *Networks*, 23 (1993) 207–210.

[12] F. Harary, Graph Theory, (Addison-Wesley, Reading, MA, 1969).

[13] L. Lovász and M.D. Plummer, Matching Theory, *Ann. Discrete Math*, 29. North-Holland Publishing Co., Amsterdam; Akadmiai Kiad (Publishing House of the Hungarian Academy of Sciences), Budapest, 1986. xxvii+544 pp.

[14] R.M. McConnell and J. P. Spinrad, Modular decomposition and transitive orientation, *Discrete Mathematics*, 201 (1999) 199–241.

[15] G.J. Minty, On maximal independent sets of vertices in claw-free graphs. *J. Combinatorial Theory*, Ser.B, 28 (1980) 284–304.

[16] D. Nakamura and A. Tamura, A revision of Minty's algorithm for finding a maximum weight stable set of a claw-free graph, *J. Oper. Res. Soc. Japan* 44 (2001) 194–204.

[17] S. Poljak, A note on stable sets and coloring of graphs, *Comment. Math. Univ. Carolinae* 15 (1974) 307–309.

[18] N. Sbihi, Algorithme de recherche d'un independent de cardinalité maximum dans un graphe sans étoile, *Discrete Mathematics*, 29 (1980) 53–76.

[19] A. Schrijver, Combinatorial Optimization, Polyhedra and Efficiency, *Series: Algorithms and Combinatorics, Vol. 24*, Springer, 2003, Chapter 69, pp. 1208-1217.