On the minimum feedback vertex set problem: Exact and enumeration algorithms*

Fedor V. Fomin[†‡] Serge Gaspers[†‡] Artem V. Pyatkin[§‡]
Igor Razgon[¶]

November 16, 2007

Abstract

We present a time $\mathcal{O}(1.7548^n)$ algorithm finding a minimum feedback vertex set in an undirected graph on n vertices. We also prove that a graph on n vertices can contain at most 1.8638^n minimal feedback vertex sets and that there exist graphs having $105^{n/10} \approx 1.5926^n$ minimal feedback vertex sets.

Keywords: *Minimum feedback vertex set, maximum induced forest, exact exponential algorithm, number of minimal feedback vertex sets*

1 Introduction

The problem of finding a minimum feedback vertex set in a graph, i.e. the smallest set of vertices whose deletion makes the graph acyclic, has many applications and its history can be traced back to the early '60s (see the survey of Festa et al. [13]). It is also one of the classical NP-complete problems from Karp's list [22]. Thus not surprisingly, for several decades, many different algorithmic approaches were tried on this problem including approximation algorithms [1, 2, 12, 23], linear programming [9], local search [4], polyhedral combinatorics [7, 20], probabilistic algorithms [26], and parameterized complexity [10, 11, 21].

The problem is approximable within a factor of 2 in polynomial time [1]. It was also extensively studied from the point of view of parameterized complexity. There was a chain of improvements (see e.g. [28]) concluding with two

*Preliminary extended abstracts of this paper appeared in the proceedings of SWAT'06 [29] and IWPEC'06 [14]

[†]Department of Informatics, University of Bergen, N-5020 Bergen, Norway. (Fedor.Fomin|Serge.Gaspers)@ii.uib.no.

[‡]Additional support by the Research Council of Norway.

[§]Sobolev Institute of Mathematics, SB RAS, 4 Acad. Koptyug avenue, 630090 Novosibirsk, Russia. The work was partially supported by grants of the Russian Foundation for Basic Research (project code 05-01-00395), INTAS (project code 04–77–7173). artem@math.nsc.ru.

[¶]Computer Science Department, University College Cork, Ireland. Supported by Science Foundation Ireland (Grant Number 05/IN/I886). i.razgon@cs.ucc.ie.

$2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$-time algorithms obtained independently by different research groups [10, 21]. It had been open for a long time whether computing a feedback vertex set of a *directed* graph is fixed-parameter tractable. Very recently this question has been resolved positively by two independent groups [8, 30].

In recent years the topic of exact (exponential-time) algorithms for NP-hard problems has led to much research (see the surveys [16, 33]). However, despite much progress on exponential-time solutions to other graph problems such as chromatic number [3, 5, 24], maximum independent set [17, 31], and minimum dominating set [15], no algorithm faster than the trivial $2^n \cdot n^{\mathcal{O}(1)}$ was known for feedback vertex set until recently. For some special graph classes, like bipartite graphs or graphs of maximum degree 4, algorithms of running time $\mathcal{O}(1.8621^n)$ and $\mathcal{O}(1.945^n)$ respectively can be found in the literature [19, 27].

The first exact algorithm breaking the trivial 2^n barrier is due to the fourth author [29]. The running time $\mathcal{O}(1.8899^n)$ of the algorithm from [29] was reduced by the first three authors in [14] to $\mathcal{O}(1.7548^n)$. Both results are based on the algorithm which can be seen as a branching algorithm (or a variation of Davis-Putnam-style exponential-time backtracking). The main idea behind breaking the 2^n barrier is based on the choice of the measure of the subproblems recursively generated by the algorithm. A good choice of the measure leads us to a significantly better worst case time analysis of the branching algorithm. The exact algorithm given in this paper that finds a minimum feedback vertex set in time $\mathcal{O}(1.7548^n)$ resulted by merging preliminary results announced in [29] and [14].

By making use of similar ideas, we show that every graph on n vertices contains at most 1.8638^n *minimal* feedback vertex sets. It is the first known upper bound for the number of minimal feedback vertex sets breaking the trivial $\mathcal{O}(2^n/\sqrt{n})$ bound (which is roughly the maximum number of subsets of an n-set such that none of them is contained in the other). This bound has algorithmic consequences as well. By the result of Schwikowski and Speckenmeyer [32], all minimal feedback vertex sets can be enumerated with polynomial time delay. Thus our result implies that the running time of the algorithm by Schwikowski and Speckenmeyer is $\mathcal{O}(1.8638^n)$. We also show that there exist graphs with at least 1.5926^n minimal feedback vertex sets.

The combinatorial bound on the number of feedback vertex sets is of independent interests. One of the very natural questions in graph theory is: how many minimal (maximal) vertex subsets satisfying a given property can be contained in a graph on n vertices? The trivial bound is $\mathcal{O}(2^n/\sqrt{n})$. Surprisingly, for very few problems better bounds, i. e. bounds of the form $\mathcal{O}(c^n)$ for $c < 2$, are known. One example of such a bound is the celebrated Moon and Moser [25] theorem stating that every graph on n vertices has at most $3^{n/3}$ maximal cliques (independent sets). Another example is the result from [18], where it is shown that the number of minimal dominating sets is at most 1.7170^n.

The rest of the paper is organized as follows. Section 2 contains preliminary results. In section 3 we present a time $\mathcal{O}(1.7548^n)$ algorithm finding a minimum feedback vertex set in a graph on n vertices. In section 4 we prove that every

graph on n vertices has at most 1.8638^n minimal feedback vertex sets and that there exists an infinite family of graphs having 1.5926^n minimal feedback vertex sets.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph on n vertices. For $V' \subseteq V$ we denote by $G[V']$ the graph induced by V' and by $G \setminus V'$ the graph induced by $V \setminus V'$. For a vertex $v \in V$ let $N(v)$ be the set of its neighbors. We denote by $\Delta(G)$ the maximum vertex degree of G.

A set $X \subseteq V$ is called a *feedback vertex set* or an FVS if $G \setminus X$ is a forest. An FVS is *minimal* if it does not contain any other FVS as a proper subset, and *minimum* if it has minimum cardinality among all FVS's in a graph. Let us note that X is a minimal (minimum) FVS if and only if $G \setminus X$ is a maximal (maximum) induced forest. Thus the problem of finding a minimum FVS is equivalent to the problem of finding a maximum induced forest or an MIF. Similarly, the number of minimal feedback vertex sets in a graph is equal to the number of maximal induced forests. For the description of the algorithm it is more convenient to work with MIF than with FVS.

We call a subset $F \subseteq V$ *acyclic* if $G[F]$ is a forest and *independent* if $G[F]$ has no edges. The notions of maximal and maximum independent sets are defined similarly to those of FVS's. If F is acyclic then every connected component of $G[F]$ on at least two vertices is called *non-trivial*.

If T is a non-trivial component then we denote by $\mathrm{Id}(T, t)$ the operation of contracting all edges of T into one vertex t and removing appeared loops. Note that this operation may create multiedges in G. We denote by $\mathrm{Id}^*(T, t)$ the operation $\mathrm{Id}(T, t)$ followed by the removal of all vertices connected with t by multiedges.

For an acyclic subset $F \subseteq V$, denote by $\mathcal{M}_G(F)$ and by $\mathcal{M}^*_G(F)$ the set of all maximal and maximum acyclic supersets of F in G, respectively (we omit the subindex G when it is clear from the context which graph is meant). Let $\mathcal{M}^* = \mathcal{M}^*(\emptyset)$. Then the problem of finding a MIF can be stated as finding an element of $\mathcal{M}^*$. We solve a more general problem, namely finding an element of $\mathcal{M}^*(F)$ for an arbitrary acyclic subset F.

To simplify the description of the algorithm, we suppose that F is always an independent set. The next proposition justifies this assumption.

Proposition 1. *Let $G = (V, E)$ be a graph, $F \subseteq V$ be an acyclic subset of vertices and T be a non-trivial component of F. Denote by G' the graph obtained from G by the operation $\mathrm{Id}^*(T, t)$ and let $F' = F \cup \{t\} \setminus T$. Then*

- *$X \in \mathcal{M}_G(F)$ if and only if $X' \in \mathcal{M}_{G'}(F')$, and*
- *$X \in \mathcal{M}^*_G(F)$ if and only if $X' \in \mathcal{M}^*_{G'}(F')$,*

where $X' = X \cup \{t\} \setminus T$.

Proof. Assume that $X \in \mathcal{M}_G(F)$. If after the operation $\mathrm{Id}(T,t)$ a vertex v is connected with t by a multiedge, then the set $T \cup \{v\}$ is not acyclic in G. Hence, no element of $\mathcal{M}_G(F)$ may contain v. In other words, X does not contain any vertices removed by the transformation from G to G' and hence $X' = X \cup \{t\} \setminus T$ is a set of vertices of G'. Moreover, X' is an acyclic subset of G'. To see this, assume by contradiction that X' induces a cycle C' in G'. Then C' necessarily includes t because otherwise C' is induced by X in G in contradiction to the acyclicity of X. Let x_1 and x_2 be two neighbors of t in C'. It follows that there is a path in G from x_1 to x_2 including vertices of T only. Replace t in C' by such a path. As a result we obtain a cycle induced by X in G in contradiction to the acyclicity of X. It remains to show that X' is a maximal acyclic subset of G'. For this purpose, assume that there is a vertex $v \in V(G') \setminus X'$ such that $X' \cup \{v\}$ is an acyclic subset. Then $X \cup \{v\}$ is an acyclic subset of G (any cycle in $X \cup \{v\}$ can be transformed into a cycle in $X' \cup \{v\}$ by the operation $\mathrm{Id}(T,t)$) larger than X in contradiction to the maximality of X.

Arguing similarly, we can prove that if $X' \in \mathcal{M}_{G'}(F')$ then $X \in \mathcal{M}_G(F)$ and that $X \in \mathcal{M}^*_G(F)$ if and only if $X' \in \mathcal{M}^*_{G'}(F')$.

□

By using the operation Id^* on every non-trivial component of F, we obtain an independent set F'.

The following proposition is used to justify the main branching rule of the algorithm.

Proposition 2. *Let $G = (V,E)$ be a graph, $F \subseteq V$ be an independent subset of vertices and $v \notin F$ be a vertex adjacent to exactly one vertex $t \in F$. Then*

1. *For every $X \in \mathcal{M}(F)$, either v or at least one vertex of $N(v) \setminus \{t\}$ is in X.*
2. *There exists $X \in \mathcal{M}^*(F)$ such that either v or at least two vertices of $N(v) \setminus \{t\}$ are in X.*

Proof. 1. If there is $X \in \mathcal{M}(F)$ such that $v \notin X$ and no vertex of $N(v) \setminus \{t\}$ is in X, then $X \cup \{v\}$ is also an induced forest of G. Thus X is not maximal, which is a contradiction.

2. Let us consider $X \in \mathcal{M}^*(F)$ such that $v \notin X$. By item 1, at least one vertex $z \in N(v) \setminus \{t\}$ is in X. For the sake of contradiction, let us assume that z is the only such vertex. Since X is maximal, we have that $X \cup \{v\}$ is not acyclic. Because v is of degree at most 2 in $G[X \cup \{v\}]$, we conclude that all the cycles in $G[X \cup \{v\}]$ must contain z. Then the set $X \cup \{v\} \setminus \{z\}$ is in $\mathcal{M}^*(F)$ and satisfies the conditions. □

Consequently, if $N(v) = \{t, v_1, v_2, \ldots, v_k\}$, then there exists $X \in \mathcal{M}^*(F)$ satisfying one of the following properties:

1. $v \in X$;

2. $v \notin X$, $v_i \in X$ for some $i \in \{1, 2, \ldots, k-2\}$ while $v_j \notin X$ for all $j < i$;

3. $v, v_1, v_2, \ldots, v_{k-2} \notin X$ but $v_{k-1}, v_k \in X$.

In particular, if $k \leq 1$, then $v \in X$ for some $X \in \mathcal{M}^*(F)$.

The following proposition is needed to handle the case where every vertex in $V \setminus F$ is adjacent to a vertex $t \in F$. We reduce this case to finding a maximal (respectively maximum) independent set in the graph $G[V \setminus F]$ with some additional edges.

Proposition 3. *Let $G = (V, E)$ be a graph and F be an independent set in G such that $V \setminus F = N(t)$ for some $t \in F$. Consider the graph $G' = G[N(t)]$ and for every pair of vertices $u, v \in N(t)$ having a common neighbor in $F \setminus \{t\}$ add an edge uv to G'. Denote the obtained graph by H and let $I \subseteq N(t)$. Then $F \cup I \in \mathcal{M}_G(F)$ if and only if I is a maximal independent set in H. In particular, $F \cup I \in \mathcal{M}^*_G(F)$ if and only if I is a maximum independent set in H.*

Proof. Let $X \in \mathcal{M}_G(F)$ and $u, v \in V \setminus F$. If $uv \in E$ then u, v, t form a triangle. If there is a vertex $w \in F \setminus \{t\}$ adjacent to both u and v then $tuwv$ is a 4-cycle. In both cases, X cannot contain u and v at the same time. Therefore, $X \in \mathcal{M}_G(F)$ if and only if $X \setminus F$ is a maximal independent set in H. □

There are several fast exponential algorithms computing a maximum independent set in a graph. We use the polynomial space algorithm of Robson.

Proposition 4 ([31])**.** *Let G be a graph on n vertices. Then a maximum independent set in G can be found in time $\mathcal{O}(1.2278^n)$.*

We need also the following well known result of Moon and Moser [25]:

Proposition 5 ([25])**.** *A graph on n vertices has at most $3^{n/3}$ maximal independent sets.*

3 Computing a minimum feedback vertex set

In this section we show how to compute the minimum size of a feedback vertex set. Our algorithm can easily be turned into an algorithm computing at least one such set. Instead of working with feedback vertex sets directly, the algorithm finds the maximum size of an induced forest in a graph. In fact, it solves a more general problem: for any acyclic set F it finds the maximum size of an induced forest containing F.

During the work of the algorithm one vertex $t \in F$ is called an *active vertex*. The algorithm branches on a chosen neighbor of t. Let $v \in N(t)$. Denote by K the set of all vertices of F other than t that are adjacent to v. Let G' be the graph obtained after the operation $\mathrm{Id}(K \cup \{v\}, u)$. We say that a vertex $w \in V \setminus \{t\}$ is a *generalized neighbor* of v in G if w is the neighbor of u in

G'. Denote by $\mathrm{gd}(v)$ the *generalized degree* of v which is the number of its generalized neighbors.

The description of the algorithm consists of a sequence of cases and subcases. To avoid a confusing nesting of if-then-else statements let us use the following convention: the first case which applies is used in the algorithm. Thus, inside a given case, the hypotheses of all previous cases are assumed to be false.

Algorithm $\texttt{mif}(G, F)$ computing for a given graph G and an acyclic set F the maximum size of an induced forest containing F is described by the following preprocessing and main procedures (let us note that $\texttt{mif}(G, \emptyset)$ computes the maximum size of an induced forest in G).

Preprocessing

1. If G consists of $k \geq 2$ connected components $G_1, G_2, \ldots, G_k$, then the algorithm is called on each of the components and

$$\texttt{mif}(G, F) = \sum_{i=1}^{k} \texttt{mif}(G_i, F_i),$$

   where $F_i = G_i \cap F$ for all $i \in \{1, 2, \ldots, k\}$.

2. If F is not independent, then apply operation $\mathrm{Id}^*(T, v_T)$ on an arbitrary non-trivial component T of F. If T contains the active vertex then v_T becomes active. Let G' be the resulting graph and let F' be the set of vertices of G' obtained from F. Then

$$\texttt{mif}(G, F) = \texttt{mif}(G', F') + |T| - 1.$$

Main procedures

1. If $F = V$ then $\mathcal{M}_G(F) = \{V\}$. Thus,

$$\texttt{mif}(G, F) = |V|.$$

2. If $F = \emptyset$ and $\Delta(G) \leq 1$ then $\mathcal{M}_G(F) = \{V\}$ and

$$\texttt{mif}(G, F) = |V|.$$

3. If $F = \emptyset$ and $\Delta(G) \geq 2$ then the algorithm chooses a vertex t in G of degree at least 2. Then t is either contained in a maximum induced forest or not. Thus the algorithm branches on two subproblems and returns the maximum:

$$\texttt{mif}(G, F) = \max \quad \{ \quad \texttt{mif}(G, F \cup \{t\}), \\ \texttt{mif}(G \setminus \{t\}, F)\}.$$

4. If F contains no active vertex then choose an arbitrary vertex $t \in F$ as an active vertex. Denote the active vertex by t from now on.

5. If $V \setminus F = N(t)$ then the algorithm constructs the graph H from Proposition 3 and computes a maximum independent set I in H. Then

$$\texttt{mif}(G, F) = |F| + |I|.$$

6. If there is $v \in N(t)$ with $\mathrm{gd}(v) \leq 1$ then add v to F.

$$\texttt{mif}(G, F) = \texttt{mif}(G, F \cup \{v\})$$

7. If there is $v \in N(t)$ with $\mathrm{gd}(v) \geq 4$ then either add v to F or remove v from G.

$$\begin{aligned}\texttt{mif}(G, F) = \max \quad \{ \quad & \texttt{mif}(G, F \cup \{v\}), \\ & \texttt{mif}(G \setminus \{v\}, F)\}\end{aligned}$$

8. If there is $v \in N(t)$ with $\mathrm{gd}(v) = 2$ then denote its generalized neighbors by w_1 and w_2. Either add v to F or remove v from G but add w_1 and w_2 to F. If adding w_1 and w_2 to F induces a cycle, we just ignore the last branch.

$$\begin{aligned}\texttt{mif}(G, F) = \max \quad \{ \quad & \texttt{mif}(G, F \cup \{v\}), \\ & \texttt{mif}(G \setminus \{v\}, F \cup \{w_1, w_2\})\}\end{aligned}$$

9. If all vertices in $N(t)$ have exactly three generalized neighbors then at least one of these vertices must have a generalized neighbor outside $N(t)$, since the graph is connected and the condition of the case Main 5 does not hold. Denote such a vertex by v and its generalized neighbors by w_1, w_2 and w_3 in such a way that $w_1 \notin N(t)$. Then we either add v to F; or remove v from G but add w_1 to F; or remove v and w_1 from G and add w_2 and w_3 to F. Similarly to the previous case, if adding w_2 and w_3 to F induces a cycle, we just ignore the last branch.

$$\begin{aligned}\texttt{mif}(G, F) = \max \quad \{ \quad & \texttt{mif}(G, F \cup \{v\}), \\ & \texttt{mif}(G \setminus \{v\}, F \cup \{w_1\}), \\ & \texttt{mif}(G \setminus \{v, w_1\}, F \cup \{w_2, w_3\})\}\end{aligned}$$

The correctness and the running time of the algorithm are analyzed in the following.

Theorem 6. *Let G be a graph on n vertices. Then a maximum induced forest of G can be found in time $\mathcal{O}(1.7548^n)$.*

Proof. Let us consider the algorithm $\texttt{mif}(G, F)$ described above. The correctness of **Preprocessing 1** and **Main 1,2,3,4,7** is clear. The correctness of **Main 5** follows from Proposition 3, while the correctness of **Preprocessing 2**

and **Main 6,8,9** follows from Proposition 1 and 2 (indeed, applying Proposition 2 to the vertex u of the graph G' shows that for some $X \in \mathcal{M}_G(F)$ either v or at least two of its generalized neighbors are in X).

In order to evaluate the time complexity of the algorithm we use the following measure:

$$\mu = |V \setminus F| + \alpha |V \setminus (F \cup N(t))|$$

where $\alpha = 0.955$. In other words, each vertex in F has weight 0, each vertex in $N(t)$ has weight 1, each other vertex has weight $1 + \alpha$, and the size of the problem is equal to the sum of the vertex weights. We will prove that a problem of size μ can be solved in time $\mathcal{O}(x^\mu)$ where

$$x < 1.33328.$$

During its execution, the algorithm naturally explores the search tree whose nodes are associated with the pairs (G, F) to which the algorithm is recursively applied. Since the algorithm spends polynomial time per node of the search tree, its running time is polynomially related to the number of nodes of the search tree.

Observe that each path from the root to a leaf of the tree is of length $\mathcal{O}(n)$. To observe this, it is sufficient to show that if (G, F) corresponds to a non-leaf node of the tree and (G_1, F_1) corresponds to a child of (G, F) then either (G_1, F_1) is a leaf or

$$|V(G_1)| + |V(G_1) \setminus F_1| \leq |V(G)| + |V(G) \setminus F| - 1.$$

For the preprocessing cases and the cases **Main 3,6,7,8,9**, this immediately follows from the description. Cases **Main 1,2** correspond to the leaf nodes. As Case **Main 4** never occurs in two consecutive nodes of the search tree, its statement may be reformulated as "choose an arbitrary vertex t as new active vertex and go through the list of cases again to select the appropriate one". That is, the node corresponding to the case **Main 4** may be analyzed together with the next node where t is specified. Finally in case **Main 5**, the graph H has exactly μ vertices since each vertex that is not in F has weight 1. By Theorem 4, a maximum independent set in H can be found in time $\mathcal{O}(1.2278^\mu)$. Thus, in the considered case we may represent (G, F) as the parent of 1.2278^μ leaves which reflect the runtime spent for processing the subtree rooted by (G, F) and hence preserve the polynomial relation of the number of nodes of the search tree to the running time.

The argumentation in the previous paragraph verifies that each path from the root to a leaf of the tree is of length $\mathcal{O}(n)$. It follows that the number of nodes of the search tree is $\mathcal{O}(n)$ multiplied by the number of leaves of the search tree. Taking into account that the running time of the algorithm is polynomially related to the number of nodes of the search tree, we obtain $T(\mu) = \mathcal{O}(f(\mu) \cdot n^{\mathcal{O}(1)})$, where $T(\mu)$ is the worst-case runtime of the algorithm called on the problem of size μ and $f(\mu)$ is the largest number of leaves of the search tree corresponding to an execution of the algorithm when applied to a problem of size

μ. We use induction on μ to prove that $f(\mu) \leq x^\mu$. Then, since the polynomial is suppressed by rounding the exponential base, we have $T(\mu) = \mathcal{O}(1.33328^\mu)$. Clearly, $f(0) = 1$. Suppose that $f(k) \leq x^k$ for every $k < \mu$ and consider a problem of size μ.

It is now clear that the following steps do not contribute to the exponential factor of the running time of the algorithm: **Preprocessing 1,2** and **Main 1,2,4,6**.

If the condition of the case **Main 5** then, by construction the number of leaves is 1.2278^μ which is smaller than 1.33328^μ.

In all the remaining cases the algorithm is called recursively on smaller problems. We consider these cases separately.

In the case **Main 3** every vertex has weight $1 + \alpha$. So, removing v leads to a problem of size $\mu - 1 - \alpha$. Otherwise, v becomes active after the next Main 4 step. Then all its neighbors become of weight 1, and we obtain a problem of size at most $\mu - 1 - 3\alpha$ since v has degree at least 2. Thus

$$f(\mu) \leq f(\mu - 1 - \alpha) + f(\mu - 1 - 3\alpha) \leq (x^{\mu-1-\alpha} + x^{\mu-1-3\alpha}) \leq x^\mu$$

by the induction assumption and the choice of x and α.

In the case **Main 7** removing the vertex v decreases the size of the problem by 1. If v is added to F then we obtain a non-trivial component in F, which is contracted into a new active vertex t' at the next Preprocessing 2 step. Those of the generalized neighbors of v that had weight 1 will be connected with t' by multiedges and thus removed during the next Preprocessing 2 step. If a generalized neighbor of v had weight $1 + \alpha$ then it will become a neighbor of t', i. e. of weight 1. Thus, in any case the size of the problem is decreased by at least $1 + 4\alpha$. So, we have that

$$f(\mu) \leq f(\mu - 1) + f(\mu - 1 - 4\alpha) \leq (x^{\mu-1} + x^{\mu-1-4\alpha}) \leq x^\mu.$$

In the case **Main 8** we distinguish three subcases depending on the weights of the generalized neighbors of v. Let i be the number of generalized neighbors of v having weight $1 + \alpha$. Adding v to F reduces the weight of a generalized neighbor either from 1 to 0 or from $1 + \alpha$ to 1. Removing v from the graph reduces the weight of both generalized neighbors of v to 0 (since we add them to F). According to this, we obtain three recurrences: for $i \in \{0, 1, 2\}$,

$$f(\mu) \leq f(\mu - (3 - i) - i\alpha) + f(\mu - 3 - i\alpha) \leq (x^{\mu-3+i-i\alpha} + x^{\mu-3-i\alpha}) \leq x^\mu.$$

The case **Main 9** is considered analogously to the case Main 8, except that at least one of the generalized neighbors of v has weight $1 + \alpha$, that is $i \geq 1$ ($i = 0$ is excluded by Main 5). In this case, we have for $i \in \{1, 2, 3\}$,

$$\begin{aligned} f(\mu) &\leq f(\mu - (4 - i) - i\alpha) + f(\mu - 2 - \alpha) + f(\mu - 4 - i\alpha) \\ &\leq (x^{\mu-4+i-i\alpha} + x^{\mu-2-\alpha} + x^{\mu-4-i\alpha}) \leq x^\mu. \end{aligned}$$

Thus

$$f(\mu) \leq x^\mu.$$

Since every vertex of G is of weight at most $1+\alpha$, we have that the running time of the algorithm is

$$T(\mu) = \mathcal{O}(x^{\mu}) = \mathcal{O}(x^{(1+\alpha)n}) = \mathcal{O}(1.33328^{1.955n}) = \mathcal{O}(1.7548^n).$$

□

The improved running time of the algorithm, compared to the algorithm in [29] is based on **Main 5** and **Main 9** and their analysis. The removal of these cases and replacing $gd(v) \geq 4$ by $gd(v) \geq 3$ in Case **Main 7** results in the $\mathcal{O}(1.8899^n)$ algorithm presented in [29].

Remark. The only tight recurrence is the one of case Main 7 when v has degree 4. Thus, an improvement of this case would improve the overall (upper bound of the) running time of the algorithm.

4 On the number of minimal feedback vertex sets

In this section we use the Branch & Reduce method in order to obtain an upper bound of 1.8638^n for the number of maximal induced forests (and thus the number of minimal feedback vertex sets) in a graph G on n vertices. It follows from the result of Schwikowski and Speckenmeyer [32] that all maximal induced forests and all minimal feedback vertex sets can be enumerated in time $\mathcal{O}(1.8638^n)$.

We also give a lower bound, namely we exhibit an infinite family of graphs, all having $105^{n/10} \approx 1.5926^n$ maximal induced forests. Thus, the worst-case running time of the algorithm in [32] is between $\mathcal{O}(1.5926^n)$ and $\mathcal{O}(1.8638^n)$.

First, we prove the upper bound for the number of maximal induced forests.

Theorem 7. *A graph G on n vertices contains at most 1.8638^n maximal induced forests.*

Proof. To prove the theorem, we show that $|\mathcal{M}_G(\emptyset)| \leq 1.8638^n$. We will prove a slightly stronger statement, namely that for any acyclic subset F of $G = (V, E)$, $|\mathcal{M}_G(F)| \leq 1.8638^n$. By Proposition 1 we may assume that F is independent. For a graph G, an independent set F and a vertex $t \in F$ (we call such a vertex t an *active vertex*), we use the same kind of measure as in the previous section:

$$\mu(G, F, t) = |V \setminus F| + \alpha|V \setminus (F \cup N(t))|,$$

where

$$\alpha = 0.5491.$$

In the case where $F = \emptyset$, we put

$$\mu(G, \emptyset) = |V|(1 + \alpha).$$

Note, that $\mu(G,F,t) \leq \mu(G,\emptyset) = (1+\alpha)n$ for every F and $t \in F$. Let $f(G,F) = |\mathcal{M}_G(F)|$ be the number of maximal induced forests containing F and let $f(\mu)$ be a maximum $f(G,F)$ among all triples (G,F,t) of measure at most μ. We claim that for $x = 1.49468$,

$$f(\mu) \leq x^{\mu}.$$

Since for $F = \emptyset$ every vertex of G has weight $1+\alpha$, the claim implies that $|\mathcal{M}_G(\emptyset)| \leq x^{(1+\alpha)n} \leq 1.49468^{1.5491n} \leq 1.8638^n$, which proves the theorem.

Let us observe that the claim is true for $\mu = 0$. In fact, for $\mu = 0$ we have that $F = V$. Thus $\mathcal{M}_G(F) = \{V\}$ and $f(0) = 1$. To prove the claim we proceed by induction assuming that $f(k) \leq x^k$ for every $k < \mu$. Let (G,F,t) be an instance of measure μ.

We consider several cases. As in the previous section, we assume that inside a given case, the hypotheses of all previous cases are assumed to be false.

Case 1: *G is not connected.* Denote by $G_1, G_2, \ldots, G_k$ the components of G. Let F_i denote the intersection of F and the vertices of G_i, for $i = 1, 2, ..., k$. If the vertices of $V \setminus F$ are present in at least two components, then, by the induction assumption,

$$f(\mu) = \prod_{i=1}^{k} f(G_i, F_i) \leq \prod_{i=1}^{k} x^{\mu(G_i,F_i)} = x^{\sum_{i=1}^{k} \mu(G_i,F_i)} = x^{\mu}.$$

Otherwise, each component which does not contain vertices of $V \setminus F$ has exactly one maximal induced forest (see the next case) and the component including all the vertices of $V \setminus F$ (which determines the overall number of the maximal induced forests) has less vertices than G. Hence we may consider that we prove the theorem by two-dimensional induction, the first dimension is the induction on μ, the second dimension is induction on the number of vertices of the underlying graph. The considered case follows from the induction assumption of the second dimension. In fact, this is the only place in the proof where the second dimension is ever used.

Case 2: $F = \emptyset$. If $\Delta(G) \leq 1$ then $\mathcal{M}_G(F) = \{V\}$, i. e. $f(G,F) = 1$. Otherwise, let t be a vertex in G of degree at least 2. Then every maximal forest either contains t, or does not. Thus the number of maximal forests in G is equal to the number of maximal forests containing t, that is $f(G,\{t\})$, plus the number of maximal forests not containing t, that is $f(G \setminus \{t\}, \emptyset)$. Since

$$\mu(G,\{t\},t) \leq \mu - 1 - 3\alpha$$

and

$$\mu(G \setminus \{t\}, \emptyset) \leq \mu - 1 - \alpha,$$

we use the induction assumption and arrive at

$$f(\mu) \leq f(\mu - 1 - 3\alpha) + f(\mu - 1 - \alpha) \leq x^{\mu-1-3\alpha} + x^{\mu-1-\alpha} \leq x^{\mu}.$$

From now on we denote by $t \in F$ an active vertex (if $F \neq \emptyset$ contains no such vertex, we may always choose an arbitrary vertex as active, reducing the measure).

Case 3: $V \setminus F = N(t)$. Then by Proposition 3, $f(\mu)$ is equal to the number of maximal independent sets in the graph H from Proposition 3. Since all vertices of $V \setminus F$ have weight 1, H has μ vertices. By Proposition 5,

$$f(\mu) \leq 3^{\mu/3} \leq x^{\mu}.$$

Now we assume that $V \setminus F \neq N(t)$, that $F \neq \emptyset$ and that G is connected. Then there is a vertex $v \in N(t)$ such that at least one of its generalized neighbors lies not in $N(t)$ (and thus contributes the weight $1+\alpha$ to the measure). Among all such vertices we choose a vertex v of minimum generalized degree. Similarly to the proof of Theorem 6, it follows from Propositions 1 and 2 that every $X \in \mathcal{M}_G(F)$ must contain either v or at least one of its generalized neighbors.

Case 4: $\mathrm{gd}(v) = 0$. In this case every $X \in \mathcal{M}_G(F)$ contains v and thus $f(G,F) = f(G, F \cup \{v\})$. Since $\mu(G, F \cup \{v\}, t) < \mu$, we have that $f(\mu) \leq x^{\mu}$.

Case 5: $\mathrm{gd}(v) = 1$. Every forest $X \in \mathcal{M}_G(F)$ either contains v, or does not contain v and contains its generalized neighbor w_1. The measure $\mu(G, F \cup \{v\}, t)$ is at most $\mu - 1 - \alpha$, and the measure $\mu(G \setminus \{v\}, F \cup \{w_1\}, t)$ is at most $\mu - 2 - \alpha$. Hence

$$f(\mu) \leq f(\mu - 1 - \alpha) + f(\mu - 2 - \alpha) \leq x^{\mu-1-\alpha} + x^{\mu-2-\alpha} \leq x^{\mu}.$$

Case 6: $\mathrm{gd}(v) = 2$. Let us denote the generalized neighbors of v by w_1 and w_2 and let us assume that $w_1 \notin N(t)$. Then every forest X from $\mathcal{M}_G(F)$

— Either contains v;

— or does not contain v and contains w_1;

— or does not contain v and w_1 but contains w_2.

Let us note that if $w_2 \in N(t)$ and v belongs to a maximal induced forest X, then w_2 does not belong to X. Thus if $w_2 \in N(t)$, then the number of forests in $\mathcal{M}(F)$ is at most

$$f(G \setminus \{w_2\}, F \cup \{v\}) + f(G \setminus \{v\}, F \cup \{w_1\}) + f(G \setminus \{v, w_1\}, F \cup \{w_2\}).$$

Thus

$$\begin{aligned} f(\mu) &\leq f(\mu - 2 - \alpha) + f(\mu - 2 - \alpha) + f(\mu - 3 - \alpha) \\ &\leq 2x^{\mu-2-\alpha} + x^{\mu-3-\alpha} \leq x^{\mu}. \end{aligned}$$

If $w_2 \notin N(t)$, then

$$\begin{aligned} f(\mu) &\leq f(\mu-1-2\alpha)+f(\mu-2-\alpha)+f(\mu-3-2\alpha) \\ &\leq x^{\mu-1-2\alpha}+x^{\mu-2-\alpha}+x^{\mu-3-2\alpha} \leq x^{\mu}. \end{aligned}$$

Case 7: $\mathrm{gd}(v) = 3$. Denote the generalized neighbors of v by w_1, w_2, and w_3 according to the rule that $w_j \notin N(t)$ and $w_k \in N(t)$ imply $j < k$. Then for every forest X from $\mathcal{M}_G(F)$ holds one of the following

— X contains v;

— X does not contain v and contains w_1;

— X does not contain v and w_1 but contains w_2;

— X does not contain v, w_1 and w_2 but contains w_3.

Let i be the number of generalized neighbors of v that are not adjacent to t. For $i = 1, 2$, we have

$$\begin{aligned} f(\mu) &\leq f(\mu-4+i-i\alpha)+f(\mu-2-\alpha)+f(\mu-3-i\alpha)+f(\mu-4-i\alpha) \\ &\leq x^{\mu-4+i-i\alpha}+x^{\mu-2-\alpha}+x^{\mu-3-i\alpha}+x^{\mu-4-i\alpha} \leq x^{\mu}. \end{aligned}$$

For $i = 3$,

$$\begin{aligned} f(\mu) &\leq f(\mu-1-3\alpha)+f(\mu-2-\alpha)+f(\mu-3-2\alpha)+f(\mu-4-3\alpha) \\ &\leq x^{\mu-1-3\alpha}+x^{\mu-2-\alpha}+x^{\mu-3-2\alpha}+x^{\mu-4-3\alpha} \leq x^{\mu}. \end{aligned}$$

Case 8: $\mathrm{gd}(v) \geq 4$. Then every forest X from $\mathcal{M}_G(F)$ either contains v or does not. Thus

$$f(\mu) \leq f(\mu-1-4\alpha)+f(\mu-1) \leq x^{\mu-1-4\alpha}+x^{\mu-1} \leq x^{\mu}.$$

□

Remark. The two tight recurrences here are in the case Main 7, when $i = 1$ and when $i = 3$. Again, an improvement of this case would provide a better bound on the number of minimal feedback vertex sets.

Now, we prove the lower bound for the number of maximal induced forests.

Theorem 8. *There exists an infinite family of graphs all having* $105^{n/10} \approx 1.5926^n$ *maximal induced forests.*

Proof. The infinite family consists of disjoint copies of the graph given in Figure 1. The same family of graphs has been used in [6] to show that the number of maximal bipartite subgraphs is lower bounded by 1.5926^n.

A *pair* of vertices in the graph of Figure 1 are two vertices whose labels differ by 5. This graph has $5 \cdot 2^4 = 80$ maximal induced forests containing one vertex

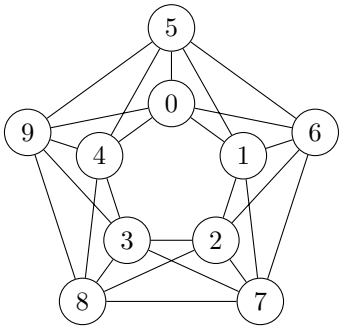

Figure 1: Generating graph for the lower bound

from 4 of the pairs, $5 \cdot 2^2 = 20$ containing one pair and one vertex from each of the opposite pairs and 5 containing two pairs. In total, it has 105 maximal induced forests.

It is clear that the maximal induced forests of a disconnected graph are the union of one maximal induced forest of each component. Their number thus equals the product of the number of maximal induced forests of each component. By taking multiple copies of the graph in Figure 1, we get the lower bound of $105^{n/10}$. □

5 Conclusion

In this paper we presented a time $\mathcal{O}(1.7548^n)$ algorithm finding a minimum feedback vertex set in an undirected graph on n vertices. We also proved that a graph on n vertices can contain at most 1.8638^n minimal feedback vertex sets and that there exist graphs having $105^{n/10} \approx 1.5926^n$ minimal feedback vertex sets. The design and analysis of algorithms establishing the first two results is based on the following three ideas. The first one is considering the complementary problem of maximum induced forest instead the straightforward computing of the feedback vertex set. The second idea is a generalization of the maximum induced problems according to which a subset of vertices F of the given graph G is introduced and the task is to find the largest forest including F as a subset. The third idea is a good choice of the measure of the subproblems recursively generated by the algorithm. This good choice led us to a significantly better worst case time analysis of the proposed algorithm.

References

[1] V. Bafna, P. Berman, and T. Fujito, *A 2-approximation algorithm for the undirected feedback vertex set problem*, SIAM Journal on Discrete Mathematics, 12 (1999), pp. 289–297.

[2] R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth, *Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference*, SIAM Journal on Computing, 27 (1998), pp. 942–959.

[3] A. Björklund and T. Husfeldt, *Inclusion-exclusion algorithms for counting set partitions*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), IEEE Computer Society, Los Alamitos, 2006, pp. 575–582.

[4] L. Brunetta, F. Maffioli, and M. Trubian, *Solving the feedback vertex set problem on undirected graphs*, Discrete Applied Mathematics, 101 (2000), pp. 37–51.

[5] J. M. Byskov, *Enumerating maximal independent sets with applications to graph colouring*, Operations Research Letters, 32 (2004), pp. 547–556.

[6] J. M. Byskov, B. A. Madsen, and B. Skjernaa, *On the number of maximal bipartite subgraphs of a graph*, J. Graph Theory, 48 (2005), pp. 127–132.

[7] M.-C. Cai, X. Deng, and W. Zang, *A min-max theorem on feedback vertex sets*, Mathematics of Operations Research, 27 (2002), pp. 361–371.

[8] J. Chen, Y. Liu, S. Lu, *Directed Feedback Vertex Set problem is FPT*, Dagstuhl Seminar Series, Seminar 07281 (2007), available electornically at `http://kathrin.dagstuhl.de/files/Materials/07/07281/07281.ChenJianer.Paper.pdf`

[9] F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson, *A primal-dual interpretation of two* 2*-approximation algorithms for the feedback vertex set problem in undirected graphs*, Operations Research Letters, 22 (1998), pp. 111–118.

[10] F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens, *An* $O(2^{O(k)}n^3)$ *FPT algorithm for the undirected feedback vertex set problem.*, in Proceedings of the 11th Annual International Conference on Computing and Combinatorics (COCOON 2005), vol. 3595 of Lecture Notes in Comput. Sci., Berlin, 2005, Springer, pp. 859–869.

[11] R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, New York, 1999.

[12] G. Even, J. Naor, B. Schieber, and L. Zosin, *Approximating minimum subset feedback sets in undirected graphs with applications*, SIAM Journal on Discrete Mathematics, 13 (2000), pp. 255–267.

[13] P. Festa, P. M. Pardalos, and M. G. C. Resende, *Feedback set problems*, in Handbook of combinatorial optimization, Supplement Vol. A, Kluwer Acad. Publ., Dordrecht, 1999, pp. 209–258.

[14] F. V. Fomin, S. Gaspers, and A. V. Pyatkin, *Finding a minimum feedback vertex set in time $O(1.7548^n)$*, in Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006), vol. 4169 of Lecture Notes in Comput. Sci., Springer, Berlin, 2006, pp. 184–191.

[15] F. V. Fomin, F. Grandoni, and D. Kratsch, *Measure and conquer: Domination – a case study*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005), vol. 3580 of Lecture Notes in Comput. Sci., Springer, Berlin, 2005, pp. 191–203.

[16] ——, *Some new techniques in design and analysis of exact (exponential) algorithms*, Bulletin of the EATCS, 87 (2005), pp. 47–77.

[17] F. V. Fomin, F. Grandoni, and D. Kratsch, *Measure and conquer: A simple $O(2^{0.288\,n})$ independent set algorithm*, in 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006), New York, 2006, ACM and SIAM, pp. 18–25.

[18] F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. Stepanov, *Bounding the number of minimal dominating sets: a measure and conquer approach*, in Proceedings of the 16th Annual International Symposium on Algorithms and Computation (ISAAC 2005), vol. 3827 of Lecture Notes in Comput. Sci., Springer, Berlin, 2005, pp. 573–582.

[19] F. V. Fomin and A. V. Pyatkin, *Finding minimum feedback vertex set in bipartite graph*, Reports in Informatics 291, University of Bergen, 2005.

[20] M. Funke and G. Reinelt, *A polyhedral approach to the feedback vertex set problem*, in Integer programming and combinatorial optimization, vol. 1084 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1996, pp. 445–459.

[21] J. Guo, R. Niedermeier, and S. Wernicke, *Parameterized complexity of generalized vertex cover problems.*, in Proceedings of the 9th International Workshop on Algorithms and Data Structures (WADS 2005), vol. 3608 of Lecture Notes in Comput. Sci., Springer, Berlin, 2005, pp. 36–48.

[22] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of computer computations, Plenum Press, New York, 1972, pp. 85–103.

[23] J. Kleinberg and A. Kumar, *Wavelength conversion in optical networks*, Journal of Algorithms, 38 (2001), pp. 25–50.

[24] M. Koivisto, *An $O(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), IEEE Computer Society, Los Alamitos, 2006, pp. 583–590.

[25] J. W. Moon and L. Moser, *On cliques in graphs*, Israel Journal of Mathematics, 3 (1965), pp. 23–28.

[26] P. M. Pardalos, T. Qian, and M. G. C. Resende, *A greedy randomized adaptive search procedure for the feedback vertex set problem*, Journal of Combinatorial Optimization, 2 (1999), pp. 399–412.

[27] V. Raman, S. Saurabh, and S. Sikdar, *Improved exact exponential algorithms for vertex bipartization and other problems*, in Proceedings of the 9th Italian Conference on Theoretical Computer Science (ICTCS 2005), vol. 3701 of Lecture Notes in Comput. Sci., 2005, pp. 375–389.

[28] V. Raman, S. Saurabh, and C. R. Subramanian, *Faster fixed parameter tractable algorithms for undirected feedback vertex set*, in Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC 2002), vol. 2518 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 2002, pp. 241–248.

[29] I. Razgon, *Exact computation of maximum induced forest*, in Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT 2006), Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 2006, pp. 160–171.

[30] I. Razgon, *Directed Feedback Vertex Set is Fixed-Parameter Tractable*, Dagstuhl Seminar Series, Seminar 07281 (2007), available electonically at `http://kathrin.dagstuhl.de/files/Submissions/07/07281/07281.RazgonIgor.Paper!.pdf`

[31] J. M. Robson, *Algorithms for maximum independent sets*, Journal of Algorithms, 7 (1986), pp. 425–440.

[32] B. Schwikowski and E. Speckenmeyer, *On enumerating all minimal solutions of feedback problems*, Discrete Appl. Math., 117 (2002), pp. 253–265.

[33] G. Woeginger, *Exact algorithms for NP-hard problems: A survey*, in Combinatorial Optimization - Eureka, you shrink!, vol. 2570 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 2003, pp. 185–207.