Finding a Minimum Feedback Vertex Set in time $\mathcal{O}(1.7548^n)$ *

Fedor V. Fomin, Serge Gaspers, and Artem V. Pyatkin**

Department of Informatics, University of Bergen,
N-5020 Bergen, Norway.
{Fedor.Fomin|Serge.Gaspers|Artem.Pyatkin}@ii.uib.no

Abstract. We present an $\mathcal{O}(1.7548^n)$ algorithm finding a minimum feedback vertex set in a graph on n vertices.

Keywords: *minimum feedback vertex set, maximum induced forest, exact exponential algorithm*

1 Introduction

The problem of finding a minimum feedback vertex set has many applications and its history can be traced back to the early '60s (see the survey of Festa at al. [2]). It is also one of the classical NP-complete problems from Karp's list [5]. There is quite a dramatic story of obtaining faster and faster parameterized algorithms with a chain of improvements (see e.g. [7]) concluding with $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$-time algorithms obtained independently by different research groups [1, 4].

A feedback vertex set of a graph on n vertices can be trivially found in time $\mathcal{O}(2^n n)$ by trying all possible vertex subsets. For a long time, despite attacks of many researchers, no faster exponential time algorithm was known. Very recently Razgon [8] broke the 2^n barrier with an $\mathcal{O}(1.8899^n)$ time algorithm. The algorithm of Razgon is based on the Branch & Reduce paradigm and its analysis is nice and clever.

In this paper we show how to find a minimum feedback vertex set in time $\mathcal{O}(1.7548^n)$. Our improvement is based on Razgon's idea of measuring the progress of the branching algorithm. The most significant improvement in the running time of our algorithm is due to a new branching rule which is based on Proposition 2. This rule works nicely except one case, which, luckily, can be reduced to finding an independent set of maximum size.

* Additional support by the Research Council of Norway.
** 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)

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 sets of its neighbors. We denote by $\Delta(G)$ the maximum vertex degree of G.

The set $X \subseteq V$ is called a *feedback vertex set* or an FVS if $G \setminus X$ is a forest. Thus the problem of finding a minimum FVS is equivalent to the problem of finding a maximum induced forest or an MIF. 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 every component of $G[F]$ is an isolated vertex; F is a *maximum independent set* of G if has a maximum cardinality among all independent sets. If F is acyclic but not independent then every connected component 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 $\mathcal{M}_G(F)$ the set of all maximum acyclic supersets of F in G (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 supposition.

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 $X' \in \mathcal{M}_{G'}(F')$ where $X' = X \cup \{t\} \setminus T$.*

Proof. 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. Therefore, the function $X \mapsto X \cup \{t\} \setminus T$ is a bijection from $\mathcal{M}_G(F)$ to $\mathcal{M}_{G'}(F')$. □

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, 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. Suppose, for the sake of contradiction, that there is $X \in \mathcal{M}(F)$ such that $v \notin X$ and only one vertex of $N(v) \setminus \{t\}$ is in X, say z. It follows from the maximality of X that $X \cup \{v\}$ is not acyclic. But since v has degree at most 2 in X all the cycles in $X \cup \{v\}$ must contain z. Then the set $X \cup \{v\} \setminus \{z\}$ is in $\mathcal{M}(F)$ and satisfies the conditions. The case where no vertex of $N(v) \setminus \{t\}$ is in X is even simpler. □

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 $X \in \mathcal{M}(F)$.

We also need the following

Proposition 3. *Let $G = (V, E)$ be a graph and F be an independent set in G such that $G \setminus F = N(t)$ 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 be a maximum independent set in H. Then $F \cup I \in \mathcal{M}_G(F)$.*

Proof. Let $X \in \mathcal{M}_G(F)$ and $u, v \in G \setminus F$. 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 maximum independent set in H. □

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

Proposition 4 ([3]). *Let G be a graph on n vertices. Then a maximum independent set in G can be found in time $\mathcal{O}(1.2210^n)$.*

3 The algorithm

In this section the algorithm finding the maximum size of an induced forest containing a given acyclic set F is presented. This algorithm can easily be turned into an algorithm computing at least one element of $\mathcal{M}_G(F)$. 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 every non-trivial component T of F. Moreover, if T contains the active vertex then v_T becomes active. Let G' be the resulting graph and let F' be the independent set in G' obtained from F. Then
$$\texttt{mif}(G, F) = \texttt{mif}(G', F') + |F \setminus F'|.$$

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 $t \in V(G)$ 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:

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

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.

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

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.

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

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.

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

The behavior of the algorithm is analyzed in the following

Theorem 1. *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

$$x < 1.333277.$$

Denote by $f(\mu)$ the maximum number of times the algorithm is called recursively on a problem of size μ (i. e. the number of leaves in the search tree). Then the running time $T(\mu)$ of the algorithm is bounded by $\mathcal{O}(f(\mu)\cdot n^{\mathcal{O}(1)})$. We use induction on μ to prove that $f(\mu)\le x^\mu$. Then $T(\mu) = \mathcal{O}(f(\mu)\cdot n^{\mathcal{O}(1)})$, and since the polynomial is suppressed by rounding the exponential base, we have $T(\mu) = \mathcal{O}(1.333277^\mu)$. Clearly, $f(0) = 1$. Suppose that $f(k)\le x^k$ for every $k<\mu$ and consider a problem of size μ.

It is 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 holds then 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 $\mathcal{O}(1.2210^{\mu})$. Also the algorithm computing a maximum independent set in [3] is a branching algorithm with a number of recursive calls bounded by $1.2210^{\mu} < 1.333277^{\mu}$.

In all 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 $i \geq 1$. 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

$$T(\mu) = \mathcal{O}(x^{\mu}) = \mathcal{O}(x^{(1+\alpha)n}) = \mathcal{O}(1.333277^{1.955n}) = \mathcal{O}(1.7548^n).$$

□

Remark 1. 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 Conclusion

We have shown that a few simple changes in the branch-and-reduce algorithm of Razgon [8] together with a flexible measure of the size of a (sub)problem leads to a significant improvement in the proved upper bound of the worst case running time of the algorithm.

Note added in camera-ready: Recently, we also proved that the number of maximal induced forests (and thus the number of minimal feedback vertex sets) in a graph on n vertices is at most 1.8638^n. Schwikowski and Speckenmeyer presented in [6] an algorithm which enumerates all minimal feedback vertex sets of a graph with polynomial time delay. Thus our upper bound implies that all minimal feedback vertex sets (and maximal induced forests) can be enumerated in time $\mathcal{O}(1.8638^n)$.

Acknowledgment. We thank Igor Razgon for sending us a preliminary version of [8].

References

1. 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 LNCS, Berlin, 2005, Springer, pp. 859–869.
2. 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.
3. 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.

4. 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 LNCS, Springer, Berlin, 2005, pp. 36–48.
5. R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of computer computations, Plenum Press, New York, 1972, pp. 85–103.
6. B. Schwikowski, and E. Speckenmeyer, *On Computing All Minimal Solutions for Feedback Problems*, in Discrete Applied Mathematics 117(1-3), 2002, pp. 253–265.
7. 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 LNCS, Springer-Verlag, Berlin, 2002, pp. 241–248.
8. I. Razgon, *Exact computation of maximum induced forest*, in Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT 2006), LNCS, Berlin, 2006, Springer, to appear.