Information Processing Letters 114 (2014) 556–560

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Faster deterministic Feedback Vertex Set

Tomasz Kociumaka, Marcin Pilipczuk *

Institute of Informatics, University of Warsaw, Poland

ARTICLE INFO

Article history:
Received 11 September 2013
Received in revised form 4 December 2013
Accepted 1 May 2014
Available online 9 May 2014
Communicated by B. Doerr

Keywords:
Algorithms
Fixed-parameter algorithm
Branching
Feedback Vertex Set

ABSTRACT

We present a new deterministic algorithm for the Feedback Vertex Set problem parameterized by the solution size. Our algorithm runs in $\mathcal{O}^*((2+\phi)^k)$ time, where $\phi < 1.619$ is the golden ratio, surpassing the previously fastest $\mathcal{O}^*((1+2\sqrt{2})^k)$-time deterministic algorithm due to Cao et al. (2010) [6]. In our development we follow the approach of Cao et al.; however, thanks to a new reduction rule, we obtain not only better dependency on the parameter in the running time, but also a solution with simple analysis and only a single branching rule.

1. Introduction

The Feedback Vertex Set problem (FVS for short), where we ask to delete as few vertices as possible from a given undirected graph to make it acyclic, is one of the fundamental graph problems, appearing on Karp's list of 21 NP-hard problems [20]. It is also one of the most-studied problems in parameterized complexity, and there has been a long 'race' for the fastest FPT algorithm parameterized by the solution size, denoted by k [2,15,16,23,19,14,18,7,6,1,12]. The fastest known *deterministic* algorithm prior to this work, due to Cao et al. [6], runs in $\mathcal{O}^*((1+2\sqrt{2})^k) \leq \mathcal{O}^*(3.83^k)$ time[1]; if we allow randomization, the Cut&Count technique yields an $\mathcal{O}^*(3^k)$ time algorithm [12]. Related research investigates kernelization complexity of FVS [5,4,25] and other variants, e.g., in directed graphs [8,13,9].

In this work we claim the lead in the 'FPT race' for the fastest *deterministic* algorithm for FVS.

Theorem 1. Feedback Vertex Set, *parameterized by the solution size k, can be solved in $\mathcal{O}^*((2+\phi)^k) \leq \mathcal{O}^*(3.619^k)$ time and polynomial space where $\phi = \frac{1+\sqrt{5}}{2} < 1.619$ is the golden ratio.*

In our developments, we closely follow the approach of the previously fastest algorithm due to Cao et al. [6]. That is, we first employ the iterative compression technique [24] in a standard manner to reduce the problem to the disjoint compression variant (Disjoint-FVS), where the vertex set is split into two parts, both inducing forests, and we are allowed to delete vertices only from the second part. Then we develop a set of reduction and branching rules to cope with this structuralized instance. We rely on the core observation of Cao et al. that the problem becomes polynomial-time solvable once the maximum degree of the deletable vertices drops to 3.

The main difference between our algorithm and the one of Cao et al. is the introduction of a new reduction rule that reduces deletable vertices with exactly one deletable neighbour and two undeletable ones. Branching on such vertices is the most costly operation in the $\mathcal{O}^*(5^k)$ time algorithm of Chen et al. [7] and avoiding such branching leads to three branching rules, with associated case analysis, in the algorithm of Cao et al. [6]. As a consequence

* Corresponding author.
E-mail addresses: kociumaka@mimuw.edu.pl (T. Kociumaka), malcin@mimuw.edu.pl (M. Pilipczuk).

[1] The $\mathcal{O}^*$-notation suppresses factors polynomial in the input size.

http://dx.doi.org/10.1016/j.ipl.2014.05.001
0020-0190/

of the new reduction rule, in our algorithm we only need a single straightforward branching rule. Thus, the new rule not only leads to a better time complexity, but also allows us to simplify the algorithm and analysis.

We also present a new and shorter proof of the main technical contribution of Cao et al., which says that Disjoint-FVS is polynomial-time solvable if all deletable vertices are of degree at most 3. Thus, apart from the better time complexity of our algorithm, we also simplify the arguments of Cao et al. [6].

1.1. Preliminaries and notation

All graphs in our work are undirected and, unless explicitly specified, simple. For a graph G, by $V(G)$ and $E(G)$ we denote its vertex- and edge-set, respectively. For $v \in V(G)$, the neighbourhood of v, denoted $N_G(v)$, is defined as $N_G(v) = \{u \in V(G) : uv \in E(G)\}$. For a set $X \subseteq V(G)$, we denote the subgraph induced by X by $G[X]$, and we use $G \setminus X$ to denote $G[V(G) \setminus X]$. For $X \subseteq V(G)$ and $v \in V(G)$ we define the X-degree of v, denoted by $\deg_X(v)$, as $|N_G(v) \cap X|$. Moreover, for $v \in X$ we say that v is X-isolated if $\deg_X(v) = 0$, and an X-leaf if $\deg_X(v) = 1$.

If $e = uv$ is an edge in a (multi)graph G such that $u \neq v$, by *contracting the edge* e we mean the following operation resulting in a multigraph G': we replace u and v with a new vertex w, and define $x' = w$ for $x \in \{u, v\}$ and $x' = x$ for $x \in V(G) \setminus \{u, v\}$. For each edge $xy \in E(G) \setminus \{e\}$, we add an edge $x'y'$ to $E(G')$. In other words, we do not suppress multiple edges and loops in the process of contraction. Moreover, there is a natural bijection between $E(G) \setminus \{e\}$ and $E(G')$. We abuse the notation and identify these edges. Note that, if G is a simple graph and $N_G(u) \cap N_G(v) = \emptyset$, no loop nor multiple edge is introduced when contracting uv.

We say that a set $F \subseteq E(G)$ is *acyclic*, if it is the edge set of a forest in G. For an acyclic set $F \subseteq E(G)$ by *contracting* F we mean a composition of subsequent contractions of edges in F, in arbitrary order. This operation is valid for every *acyclic* set $F \subseteq E(G)$ and the resulting multigraph H does not depend on the order of edge contractions.

Observation 2. *Let $F \subseteq E(G)$ be an acyclic in a multigraph G, and let a multigraph H be obtained from G by contracting F. Then any set $F' \subseteq E(G) \setminus F$ is acyclic in H if and only if $F' \cup F$ is acyclic in G.*

2. The algorithm

2.1. Iterative compression

Following the approach of Cao et al. [6], we employ the iterative compression technique [24] in a standard manner. Consider the following variant of FVS.

Disjoint-FVS
Input: A graph G, a partition $V(G) = U \cup D$ such that both $G[U]$ and $G[D]$ are forests, and an integer k.
Question: Does there exist a set $X \subseteq D$ of size at most k such that $G \setminus X$ is a forest?

A D-isolated vertex of degree 3 is called a *tent*. For a Disjoint-FVS instance $I = (G, U, D, k)$ we define the following functions: $k(I) = k$, $\ell(I)$ is the number of connected components of $G[U]$, $t(I)$ is the number of tents in I, and $\mu(I) = k(I) + \ell(I) - t(I)$ is the *measure* of I. Note that our measure differs from the one used by Cao et al. For each function we omit the argument if the instance is clear from the context.

In the rest of the paper we focus on solving Disjoint-FVS, and proving the following theorem.

Theorem 3. Disjoint-FVS *on an instance I can be solved in $\mathcal{O}^*(\phi^{\max(0,\mu(I))})$ time and polynomial space.*

For sake of completeness, we show how Theorem 3 implies Theorem 1.

Proof of Theorem 1. Assume we are given an FVS instance (G, k). Let $v_1, \ldots, v_n$ be an arbitrary ordering of $V(G)$. Define $V_i = \{v_1, v_2, \ldots, v_i\}$, and $G_i = G[V_i]$; we iteratively solve FVS instances (G_i, k) for $i = 1, 2, \ldots, n$. Clearly, if (G_i, k) turns out to be a NO-instance for some i, (G, k) is a NO-instance as well. On the other hand, (G_i, k) is a trivial YES-instance for $i \leq k + 1$.

To finish the proof we need to show how, given a solution X_{i-1} to (G_{i-1}, k), to solve the instance (G_i, k). Let $Z = X_{i-1} \cup \{v_i\}$ and $D = V_i \setminus Z = V_{i-1} \setminus X_{i-1}$. Clearly, $G[D]$ is a forest. We branch into $2^{|Z|} \leq 2^{k+1}$ subcases, guessing the intersection of the solution to (G_i, k) with the set Z. In a branch where we guess $Y \subseteq Z$, we delete Y from G_i and disallow deleting vertices of $Z \setminus Y$. More formally, for any $Y \subseteq Z$ such that $G_i[Z \setminus Y]$ is a forest, we define $U = Z \setminus Y$ and apply the algorithm of Theorem 3 to the Disjoint-FVS instance $I_Y = (G_i \setminus Y, U, D, k - |Y|)$. Clearly, I_Y is a YES-instance of Disjoint-FVS if and only if (G_i, k) has a solution X_i with $X_i \cap Z = Y$.

As for the running time, note that $\ell(I_Y) \leq |U| = |Z \setminus Y| \leq (k + 1) - |Y|$. Hence, $\mu(I_Y) \leq 2(k - |Y|) + 1$ and the total running time of solving (G_i, k) is bounded by

$$\mathcal{O}^*\left(\sum_{Y \subseteq Z} \phi^{2(k-|Y|)+1}\right) = \mathcal{O}^*\left(\left(1 + \phi^2\right)^k\right) = \mathcal{O}^*\left((2 + \phi)^k\right).$$

This completes the proof of Theorem 1. □

2.2. Reduction rules

Assume we are given a Disjoint-FVS instance $I = (G, U, D, k)$. We first state the following three reduction rules, which in the current or slightly modified version were used in previous algorithms for FVS [6,7]. At any time, we apply the lowest-numbered applicable rule first.

Reduction Rule 1. Remove all vertices of degree at most 1 from G.

Reduction Rule 2. If a vertex $v \in D$ has at least two neighbours in the same connected component of $G[U]$, delete v and decrease k by one.

Reduction Rule 3. If there exists a vertex $v \in D$ of degree 2 in G, move it to U if it has a neighbour in U, or contract one of its incident edges otherwise.