Improved Analysis of Highest-Degree Branching for Feedback Vertex Set

Yoichi Iwata
National Institute of Informatics, Tokyo, Japan
yiwata@nii.ac.jp

Yusuke Kobayashi
Kyoto University, Kyoto, Japan
yusuke@kurims.kyoto-u.ac.jp

Abstract

Recent empirical evaluations of exact algorithms for FEEDBACK VERTEX SET have demonstrated the efficiency of a highest-degree branching algorithm with a degree-based pruning heuristic. In this paper, we prove that this empirically fast algorithm runs in $O(3.460^k n)$ time, where k is the solution size. This improves the previous best $O(3.619^k n)$-time deterministic algorithm obtained by Kociumaka and Pilipczuk (Inf. Process. Lett., 2014).

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Feedback Vertex Set, Branch and bound, Measure and conquer

Digital Object Identifier 10.4230/LIPIcs.IPEC.2019.22

Funding *Yoichi Iwata*: Supported by JSPS KAKENHI Grant Number JP17K12643.
Yusuke Kobayashi: Supported by JSPS KAKENHI Grants Number JP16K16010, 17K19960, and 18H05291.

Acknowledgements We would like to thank Yixin Cao for valuable discussions and thank organizers of PACE challenge 2016 for motivating us to study FVS.

1 Introduction

FEEDBACK VERTEX SET (FVS) is a problem of finding the minimum-size vertex deletion set to make the input graph a forest, which is one of the Karp's 21 NP-complete problems [18]. It is known that this problem is *fixed-parameter tractable (FPT)* parameterized by the solution size k [2, 10]; i.e., we can find a deletion set of size k in $O^*(f(k))$[1] time for some function f. FVS is one of the most comprehensively studied problems in the field of parameterized algorithms, and various FPT algorithms using different approaches have been developed, including short-cycle branching [11], highest-degree branching [4], iterative-compression branching [5, 6, 20], LP-guided branching [16, 17], cut-and-count dynamic programming [7], and random sampling [1].

The current fastest deterministic FPT algorithm for FVS is a branching algorithm combined with the iterative compression technique [20] which runs in $O^*(3.619^k)$ time. When allowing randomization, the current fastest one is a cut-and-count dynamic programming algorithm [7] which runs in $O^*(3^k)$ time. In this paper, we give a faster deterministic algorithm which runs in $O^*(3.460^k)$ time. As explained below, this study is strongly motivated by

[1] The $O^*(\cdot)$ notation hides factors polynomial in n. Note that for FVS, any $O(f(k)n^{O(1)})$-time FPT algorithms can be improved to $O(f(k)k^{O(1)} + k^{O(1)}n)$ time by applying a linear-time polynomial-size kernel [15] as a preprocess. We can therefore focus only on the $f(k)$ factor when comparing the running time.

Parameterized Algorithms and Computational Experiments (PACE) challenge and its follow-up empirical evaluation by Kiljan and Pilipczuk [19]. Instead of designing a new theoretically fast algorithm, we analyze the theoretical worst-case running time of an empirically fast algorithm that has been developed through the PACE challenge and the empirical evaluation, and we show that this algorithm is not only empirically fast but also theoretically fast.

PACE challenge is an annual programming challenge started in 2016. Due to the importance of FVS in the field, FVS was selected as the subject of track B in the first PACE challenge [9]. Although in theoretical studies, the current fastest algorithm is the randomized cut-and-count dynamic programming, the result of the challenge suggests that branching is the best choice in practice. This is not so surprising; because the theoretical analysis of branching algorithms is difficult, the proved upper bound of the running time is not so tight, and moreover, the worst-case instances for branching algorithms are usually very rare in practice.

Among seven submissions to the PACE challenge, top six submissions used branching algorithms; the first used an LP-guided branching; the second and third used branching on highest-degree vertices; the fourth and sixth used branching combined with iterative compression; and the fifth used branching on short cycles. In the branching algorithms, we recursively solve the problem by picking a vertex v and branching into two cases: deleting v (the case when v is contained in the solution) or making v undeletable (the case when v is not contained in the solution). Because a forest has average degree less than one, in order to obtain a solution of size k, the graph must contain k high-degree deletable vertices. Based on this observation, in addition to the pruning by the LP lower bound, the first-place solver by Imanishi and Iwata [14] used the following *degree-based pruning* heuristic:

▶ **Lemma 1** ([14]). *Given a graph $G = (V, E)$ and a set of undeletable vertices $F \subseteq V$, let $v_1, v_2, \ldots$ be the vertices of $V \setminus F$ in the non-decreasing order of the degrees $d(v_i)$ in G. If $k \leq |V \setminus F|$ and $|E| - \sum_{i=1}^{k} d(v_i) \geq |V| - k$ holds, there is no feedback vertex set $S \subseteq V \setminus F$ of size k.*

The follow-up empirical evaluation [19] shows that the use of the degree-based pruning is much more important than the choice of branching rules. By combining with the degree-based pruning, the performances of the LP-guided branching [16], the highest-degree branching [4], and the iterative-compression branching [20], are all significantly improved, and among them, the highest-degree branching slightly outperforms the others. Cao [4] showed that one can stop the highest-degree branching at depth $3k$ by using a degree-based argument, and therefore the running time is $O(8^k)$. On the other hand, the theoretically proved running time of other branching algorithms (without the degree-based pruning) are $O^*(4^k)$ for the LP-guided branching [16] and $O^*(3.619^k)$ for the iterative-compression branching [20]. These affairs motivated us to refine the analysis of the highest-degree branching with the degree-based pruning.

In our analysis, instead of bounding the depth of the search tree as Cao [4] did, we design a new measure to bound the size of the search tree. The measure is initially at most k and we show that the measure drops by some amount for each branching. In contrast to the standard analysis of branching algorithms, our measure has a negative term and thus can have negative values; however, we show that we can immediately apply the degree-based pruning for all such cases. A simple analysis already leads to an $O^*(4^k)$-time upper bound which significantly improves the $O^*(8^k)$-time upper bound obtained by Cao [4]. We then apply the computer-aided measure-and-conquer analysis [12] and improve the upper bound to $O^*(3.460^k)$.

Algorithm 1 Highest-degree branching algorithm with a degree-based pruning.

```
1: if $k < 0$ then return No.
2: if $G$ is a forest then return Yes.
3: Apply reduction rules 1–6.
4: Apply the degree-based pruning.
5: Apply the highest-degree branching.
```

1.1 Organization

Section 2 describes the highest-degree branching algorithm with the degree-based pruning. In Section 3, we analyze the running time of the algorithm. We first give a simple analysis in Section 3.1 and then give a computer-aided measure-and-conquer analysis in Section 3.2. While the correctness of the simple analysis can be easily checked, we need to verify thousands of inequalities to check the correctness of the measure-and-conquer analysis. The source code of a program to verify the inequalities is available at `https://github.com/wata-orz/FVS_analysis`.

2 Algorithm

Because our algorithm may introduce multiple edges connecting the same pair of vertices, we deal with multi-graphs. Note that a double edge is also considered as a cycle. Let $G = (V, E)$ be a multi-graph. We define the degree $d(u)$ of vertex u as the total multiplicity of edges incident to u. For subset $U \subseteq V$, $G[U] = (U, E[U])$ denotes the induced subgraph, where $E[U]$ is the subset of E whose two endpoints are in U. For $U \subseteq V$, the contraction of U into a new vertex u is the following operation. We first introduce a new vertex u. For each edge $vw \in E$ with $v \in U$ and $w \notin U$, we insert an edge uw of the same multiplicity (if uw has been already inserted, its multiplicity is increased by the multiplicity of vw). Finally, we delete U and its incident edges from the graph. Note that the contraction does not change degrees of vertices in $V \setminus U$.

Algorithm 1 describes our algorithm. An input to our branching algorithm is a tuple (G, F, k) consisting of a multi-graph $G = (V, E)$, a set of undeletable vertices $F \subseteq V$ such that $G[F]$ forms a forest, and an integer k. Our task is to find a subset of vertices $S \subseteq V \setminus F$ such that $|S| \leq k$ and $G[V \setminus S]$ contains no cycles. For convenience, we use D to denote $\max_{v \in V \setminus F} d(v)$ when G and F are clear from the context (when using D, $V \setminus F$ is ensured to be non-empty). If $k < 0$, we return No, and if G is a forest, we return Yes. Otherwise, we apply reduction rules, the degree-based pruning rule, and the highest-degree branching rule. In the following, we give details of each rule with proof of correctness.

Our algorithm uses exactly the same set of reduction rules used in the empirical evaluation by Kiljan and Pilipczuk [19] listed below in the given order (i.e., rule i is applied only when none of the rules j with $j < i$ are applicable). Recall that D denotes $\max_{v \in V \setminus F} d(v)$.

▶ **Reduction Rule 1.** If there exists a vertex u of degree at most one, delete u.

▶ **Reduction Rule 2.** If there exists a vertex $u \notin F$ such that $G[F \cup \{u\}]$ contains a cycle, delete u and decrease k by one.

▶ **Reduction Rule 3.** If there exists a vertex u of degree two, add an edge connecting the two neighbors of u, and then delete u.

▶ **Reduction Rule 4.** If there exists an edge e of multiplicity more than two, reduce its multiplicity to two.

▶ **Reduction Rule 5.** If there exists a vertex $u \notin F$ incident to a double edge uw with $d(w) \leq 3$, delete u and decrease k by one.

▶ **Reduction Rule 6.** If $D \leq 3$, solve the problem in polynomial time by a reduction to the linear (co-graphic) matroid parity [5, 20].

Note that, in reduction rules 1 and 3, the vertex u might be in F, and in such a case, the reductions also delete u from F. The first four reduction rules are standard and have been used in branching FPT algorithms [4–6, 11, 20], the random sampling FPT algorithm [1], and also polynomial kernels [3, 15] for FVS. Reduction rule 5 was used in the Imanishi–Iwata solver [14] and the empirical evaluation [19], and its correctness can be proved as follows. Because there is a double edge uw, any feedback vertex set must contain at least one of u and w. Because w has at most one edge other than the double edge uw, every cycle containing w also contains u. Therefore, there always exists a minimum feedback vertex set containing u. Reduction rule 6 was introduced by Cao, Chen, and Liu[5] and then was simplified by Kociumaka and Pilipczuk [20].

▶ **Lemma 2.** *After applying the reductions, the following conditions hold.*
1. *G has minimum degree at least three.*
2. *No double edges are incident to F.*
3. *$D \geq 4$.*
4. *For any vertex $v \notin F$, $G - v$ has minimum degree at least two.*

Proof. The first three conditions clearly hold. We show the fourth condition. Suppose that in $G - v$, a vertex u has degree at most one. Because u has degree at least three in G, the set of incident edges to u in G must be a double-edge uv and a single-edge uw for another vertex w. If $u \notin F$, we can apply reduction rule 5, which is a contradiction. If $u \in F$, the double edge uv is incident to F in G, which is also a contradiction. Therefore, when no reductions are applicable, $G - v$ contains no such vertex u. ◀

If none of the reductions are applicable, we apply the following pruning rule.

▶ **Pruning Rule.** If $kD < \sum_{v \in F}(d(v) - 2)$, return NO.

The correctness of the pruning rule follows from the following lemma.

▶ **Lemma 3.** *If the minimum degree of G is at least two and $kD < \sum_{v \in F}(d(v) - 2)$ holds, there is no feedback vertex set $S \subseteq V \setminus F$ of size at most k.*

Proof. Suppose that there is a feedback vertex set $S \subseteq V \setminus F$ of size at most k. We have

$$\begin{aligned} kD - \sum_{v \in F}(d(v) - 2) &\geq \sum_{v \in S} d(v) - \sum_{v \in V \setminus S}(d(v) - 2) \\ &= \sum_{v \in S} d(v) - \sum_{v \in V \setminus S} d(v) + 2|V \setminus S| \\ &= 2|E[S]| - 2|E[V \setminus S]| + 2|V \setminus S| \\ &\geq -2|E[V \setminus S]| + 2|V \setminus S|. \end{aligned}$$

Because $G - S$ is a forest, this must be non-negative. ◀

Note that this pruning is different from the degree-based pruning (Lemma 1) used in the Imanishi–Iwata solver [14] and the empirical evaluation [19]; however, as the following lemma shows, this pruning is applicable only if the original degree-based pruning is applicable. Therefore, we can use the same analysis against the original degree-based pruning. We use this weaker version because it is sufficient for our analysis. We leave whether the stronger version helps further improve the analysis as future work.

▶ **Lemma 4.** *For a subset $F \subseteq V$, let $v_1, v_2, \dots$ be the vertices of $V \setminus F$ in the non-increasing order of the degrees $d(v_i)$ in G. If $k \leq |V \setminus F|$ and the minimum degree of G is at least two, $kd(v_1) < \sum_{v\in F}(d(v) - 2)$ implies $|E| - \sum_{i=1}^{k} d(v_i) \geq |V| - k$.*

Proof. Let $X = \{v_1, \dots, v_k\}$ and assume that $kd(v_1) < \sum_{v\in F}(d(v) - 2)$ holds. We have

$$\begin{aligned} 2\left(|V| - k - |E| + \sum_{v\in X} d(v)\right) &= \sum_{v\in X} d(v) - \left(2|E| - \sum_{v\in X} d(v) - 2(|V| - k)\right) \\ &= \sum_{v\in X} d(v) - \sum_{v\in V\setminus X} (d(v) - 2) \\ &\leq kd(v_1) - \sum_{v\in F}(d(v) - 2) < 0. \end{aligned}$$ ◀

Finally, when none of the reduction rules nor the pruning rule are applicable, we apply the highest-degree branching rule.

▶ **Branching Rule.** Pick a vertex $u \in V \setminus F$ of the highest degree D. Let U be the set of neighbors of u that are contained in F and let G' be the graph obtained by contracting $U \cup \{u\}$ into a new vertex u'. We branch into two cases: $(G - u, F, k - 1)$ and $(G', F - U + u', k)$.

We pick a vertex $u \in V \setminus F$ of the highest degree D and branch into two cases: whether u is contained in the solution or not. In the former case, we delete u and decrease k by one, and in the latter case, we make u undeletable by inserting it into F. Because no feedback vertex set $S \subseteq V \setminus F$ can delete edges inside F, if u has neighbors U in F, we can safely contract $U \cup \{u\}$. We use the contraction for simplicity of analysis; instead of using the number of connected components of $G[F]$, we can simply use $|F|$. Note that, by using an additional reduction rule, we can ensure that F forms an independent set; however, we do not use it because our analysis do not require such a strong condition.

▶ **Proposition 5.** *Algorithm 1 correctly solves* FVS.

3 Analysis

For parameters $0 \leq \alpha \leq 1$ and $(\beta_d)_{d=0}^{\infty}$ satisfying $0 = \beta_0 = \beta_1 = \beta_2 \leq \beta_3 \leq \beta_4 \leq \cdots$, we define

$$\mu(G, F, k) := k - \frac{\alpha}{D} \sum_{v\in F}(d(v) - 2) + \sum_{v\in F} \beta_{d(v)}.$$

Initially, the algorithm is called with $F = \emptyset$, and we have $\mu(G, \emptyset, k) = k$.

▶ **Lemma 6.** *If none of the reduction rules 1–6 nor the pruning rule are applicable for an input (G, F, k), then $\mu(G, F, k) \geq 0$ holds.*

Proof. If the pruning is not applicable, we have $kD \geq \sum_{v \in F}(d(v)-2)$. Hence we have

$$\mu(G,F,k) = (1-\alpha)k + \frac{\alpha}{D}\left(kD - \sum_{v\in F}(d(v)-2)\right) + \sum_{v \in F}\beta_{d(v)} \geq 0. \quad ◀$$

We now show that applying the reduction rules does not increase μ. We can easily see that D never increases by the reduction but may decrease; however, because such decrease leads to a smaller μ, we can analyze as if D is not changed by the reduction.

From condition 4 in Lemma 2 and the fact that contraction does not change degrees, the graph immediately after branching has minimum degree at least two. Hence, we apply reduction rule 1 only after applying reduction rules 2 or 5 (or against the initial instance, in which case, μ does not increase because F remains empty).

▶ **Lemma 7.** *Reduction rules 2 or 5, together with the subsequent applications of reduction rule 1, do not increase μ.*

Proof. For convenience of analysis, when a vertex of degree one is generated, we immediately put it into F. This does not affect the behavior of the algorithm because all such vertices will be deleted by reduction rule 1.

We first show that applying reduction rule 1 does not increase μ. When $d(u)=0$, we have $\mu(G-u, F-u, k) = \mu(G,F,k) - \frac{\alpha}{D}$. When $d(u)=1$, let v be the neighbor of u. If $v \in F$, we have $\mu(G-u,F-u,k) = \mu(G,F,k) + \beta_{d(v)-1} - \beta_{d(v)} \leq \mu(G,F,k)$. If $v \notin F$ and $d(v) \geq 3$, we have $\mu(G-u,F-u,k) = \mu(G,F,k) - \frac{\alpha}{D}$. If $v \notin F$ and $d(v)=2$, deleting u puts v into F, and hence we have $\mu(G-u,F-u+v,k) = \mu(G,F,k)$.

We next show that applying reduction rules 2 or 5 does not increase μ. Let P be the set of vertices in $V \setminus (F \cup \{u\})$ whose degree in $G-u$ is one. Let q be the number (i.e., the total multiplicity) of edges between u and F. Because P is a subset of neighbors of u, we have $|P| + q \leq d(u) \leq D$. Then, we have $\mu(G-u, F\cup P, k-1) \leq \mu(G,F,k) - 1 + \frac{\alpha}{D}(|P|+q) \leq \mu(G,F,k) - 1 + \alpha \leq \mu(G,F,k)$. ◀

Because reduction rule 3 deletes a vertex of degree two and does not change the degrees of other vertices, it does not change μ. Because reduction rule 4 is applied only when reduction rule 2 cannot be applied, it does not change the degrees of vertices in F, and therefore it does not change μ.

Finally, we analyze the branching rule. Because contraction does not change degrees of other vertices, D never increases by the branching but may decrease. As we did in the analysis of the reduction rules, we analyze as if D does not change by the branching. Recall that U is the set of neighbors of u that are contained in F. Let $f := |U|$ and $\mathbf{d} := \{d_1, \ldots, d_f\}$ be the multiset of degrees of vertices in U. The degree of u' is $d' := D + \sum_{i=1}^{f}(d_i - 2)$. In the former case of the branching, we have

$$\begin{aligned}\Delta_1(D,\mathbf{d}) &:= \mu(G,F,k) - \mu(G-u,F,k-1)\\ &= 1 - f\frac{\alpha}{D} + \sum_{i=1}^{f}(\beta_{d_i} - \beta_{d_i - 1}).\end{aligned}$$

In the latter case, we have

$$\begin{aligned}\Delta_2(D,\mathbf{d}) &:= \mu(G,F,k) - \mu(G',F-U+u',k)\\ &= -\frac{\alpha}{D}\left(\sum_{i=1}^{f}(d_i-2)-(d'-2)\right)+\sum_{i=1}^{f}\beta_{d_i}-\beta_{d'}\\ &= \alpha - 2\frac{\alpha}{D}+\sum_{i=1}^{f}\beta_{d_i}-\beta_{d'}.\end{aligned}$$

Now, we can prove the running time of the algorithm by induction on the height of the search tree as follows. Suppose that the number of leaves in the search tree is bounded by $c^{\mu(G,F,k)}$ for some $c > 1$ when the height is at most h. After the branching, we can bound the total number of leaves by $c^{\mu(G,F,k)-\Delta_1(D,\mathbf{d})} + c^{\mu(G,F,k)-\Delta_2(D,\mathbf{d})}$. Hence, in the induction step for $h+1$, we need to show the inequality

$$c^{\mu(G,F,k)-\Delta_1(D,\mathbf{d})} + c^{\mu(G,F,k)-\Delta_2(D,\mathbf{d})} \le c^{\mu(G,F,k)},$$

which is equivalent to

$$c^{-\Delta_1(D,\mathbf{d})} + c^{-\Delta_2(D,\mathbf{d})} \le 1.$$

Therefore, if $c^{-\Delta_1(D,\mathbf{d})} + c^{-\Delta_2(D,\mathbf{d})} \le 1$ holds for any $(D,\mathbf{d})$ for some $c > 1$, the running time of the algorithm is bounded by $O^*(c^k)$. We now optimize the parameters to minimize c.

The design of our measure

We have reached to our measure by the following three steps.

Step 1. Let gap $:= kD - \sum_{v\in F}(d(v)-2)$, i.e., the distance to the application of the pruning rule. By analyzing the branching rule, we observe that deleting u decreases k but may not decrease gap, while making u undeletable does not change k but decreases gap. Hence, by taking the convex combination $(1-\alpha)\cdot k + \alpha \cdot \frac{1}{D}$gap, we can ensure that the measure always decreases. The coefficient $\frac{1}{D}$ is for bounding the measure by k.

Step 2. We observe that deleting u decreases gap if u has neighbors not in F. When u has more than two neighbors in F, making u undeletable decreases the size $|F|$. Because $|F|$ cannot be negative, the worst-case (when all the neighbors of u are contained in F) cannot occur frequently. Hence, by taking $|F|$ into the measure, we can improve the analysis. This corresponds to the simple analysis presented in Section 3.1.

Step 3. We observe that applying reduction rule 3 against a vertex in F does not change gap but decreases $|F|$. Deleting u in the branching rule decreases the degree of its neighbors, and if a neighbor is in F, such a decrease leads to a future application of reduction rule 3. Hence, by using the sum of weights $\sum_{v\in F}\beta_{d(v)}$ depending on the degree instead of the size $|F| = \sum_{v\in F} 1$, we can improve the analysis. This corresponds to the computer-aided analysis presented in Section 3.2.

3.1 Simple Analysis

As a simple analysis whose correctness can be easily checked, we use $\alpha = \log_4 \frac{8}{3} \approx 0.7075$, $\beta_d = \frac{1}{2}\log_4 \frac{3}{2} \approx 0.1462$ for all $d \geq 3$, and $c = 4$. Note that, when applying the branching rule, $D \geq 4$ holds from Lemma 2. For these parameters, we have

$$c^{-\Delta_1(D,\mathbf{d})} \leq 4^{-1+\frac{f}{D}\log_4 \frac{8}{3}} = \frac{1}{4}\cdot\left(\frac{8}{3}\right)^{\frac{f}{D}} \leq \frac{1}{4}\min\left(\frac{8}{3}, \left(\frac{8}{3}\right)^{\frac{f}{4}}\right),$$

$$c^{-\Delta_2(D,\mathbf{d})} = 4^{\left(\frac{2}{D}-1\right)\log_4 \frac{8}{3}+(1-f)\frac{1}{2}\log_4 \frac{3}{2}} \leq \left(\frac{3}{8}\right)^{\frac{1}{2}}\cdot\left(\frac{3}{2}\right)^{\frac{1-f}{2}}.$$

We now show that $c^{-\Delta_1(D,\mathbf{d})} + c^{-\Delta_2(D,\mathbf{d})} \leq 1$ holds by the following case analysis.

$$c^{-\Delta_1(D,\mathbf{d})} + c^{-\Delta_2(D,\mathbf{d})} \leq \begin{cases} \frac{1}{4} + \left(\frac{3}{8}\right)^{\frac{1}{2}}\cdot\left(\frac{3}{2}\right)^{\frac{1}{2}} = 1 & (f=0), \\ \frac{1}{4}\cdot\left(\frac{8}{3}\right)^{\frac{1}{4}} + \left(\frac{3}{8}\right)^{\frac{1}{2}} < 0.932 & (f=1), \\ \frac{1}{4}\cdot\left(\frac{8}{3}\right)^{\frac{2}{4}} + \left(\frac{3}{8}\right)^{\frac{1}{2}}\cdot\left(\frac{2}{3}\right)^{\frac{1}{2}} < 0.909 & (f=2), \\ \frac{1}{4}\cdot\left(\frac{8}{3}\right)^{\frac{3}{4}} + \left(\frac{3}{8}\right)^{\frac{1}{2}}\cdot\frac{2}{3} < 0.930 & (f=3), \\ \frac{1}{4}\cdot\frac{8}{3} + \left(\frac{3}{8}\right)^{\frac{1}{2}}\cdot\left(\frac{2}{3}\right)^{\frac{3}{2}} = 1 & (f\geq 4). \end{cases}$$

▶ **Theorem 8.** *Algorithm 1 solves* FVS *in $O^*(4^k)$ time.*

3.2 Measure-and-Conquer Analysis

We use the parameters $\alpha = 0.922863$, β shown in Table 1, and $c = 3.460$. These values are obtained by solving a convex optimization problem to minimize c under the constraints of $c^{-\Delta_1(D,\mathbf{d})} + c^{-\Delta_2(D,\mathbf{d})} \leq 1$ for any $(D, \mathbf{d})$. For details of computer-aided measure-and-conquer analysis and how to solve it as a convex optimization problem, see [13, Chapter 2]. For $d \geq d_{\max} := 30$, we fix $\beta_d = \beta_{d_{\max}}$. This is for bounding the number of vectors $\mathbf{d}$ we need to consider. The running time for solving the convex optimization problem depends on the choice of $d_{\max}$, and we chose $d_{\max} := 30$ because increasing it to 50 did not improve the analysis, and when increasing it to 60, the optimization did not finish within an hour.

▶ **Lemma 9.** $c^{-\Delta_1(D,\mathbf{d})} + c^{-\Delta_2(D,\mathbf{d})} \leq 1$ *holds for any* $(D, \mathbf{d})$ *with* $D \geq 4$.

Proof. Suppose that $d_j \geq 32$ holds for some j. Because $\beta_d = \beta_{30}$ for all $d \geq 30$ and because $d' = D + \sum_i (d_i - 2) \geq D + (d_j - 2) \geq 31$ holds, decreasing d_j by one does not change $\Delta_1(D, \mathbf{d})$ nor $\Delta_2(D, \mathbf{d})$. Therefore, we can focus on the case of $d_i \leq 31$ for all i. We now show that the inequality holds by induction on D. When $D = 4$, we can verify that

$$c^{-\Delta_1(4,\mathbf{d})} + c^{-\Delta_2(4,\mathbf{d})} \leq 1 \ (\forall \mathbf{d}) \tag{1}$$

holds by naively enumerating all the possible configurations of $\mathbf{d}$. Assume that, for a fixed D, the inequality holds for any $(D-1, \mathbf{d})$. We show that the inequality also holds for any $(D, \mathbf{d})$.

Table 1 The values of β.

d	β_d	d	β_d	d	β_d
1	0.000000	11	0.384771	21	0.462794
2	0.000000	12	0.396884	22	0.466788
3	0.114038	13	0.408715	23	0.470435
4	0.186479	14	0.418855	24	0.473778
5	0.238143	15	0.427643	25	0.476853
6	0.277239	16	0.435333	26	0.479691
7	0.308030	17	0.442118	27	0.482320
8	0.332974	18	0.448149	28	0.484760
9	0.353536	19	0.453544	29	0.487032
10	0.370540	20	0.458401	≥ 30	0.489153

When $f < D$, we have

$$\Delta_1(D, \mathbf{d}) \geq \Delta_1(D-1, \mathbf{d})$$

and

$$\Delta_2(D, \mathbf{d}) = \Delta_2(D-1, \mathbf{d}) - 2\frac{\alpha}{D} - \beta_{d'} + 2\frac{\alpha}{D-1} + \beta_{d'-1},$$

where $d' = D + \sum_{i=1}^{f}(d_i - 2)$. When $d' > 31$, we have $\Delta_2(D, \mathbf{d}) = \Delta_2(D-1, \mathbf{d}) - 2\frac{\alpha}{D} + 2\frac{\alpha}{D-1} \geq \Delta_2(D-1, \mathbf{d})$. When $d' \leq 31$, we can verify that our parameters satisfy

$$-2\frac{\alpha}{D} - \beta_{d'} + 2\frac{\alpha}{D-1} + \beta_{d'-1} \geq 0 \ \ (\forall (D, d') \text{ with } 5 \leq D \leq d' \leq 31). \tag{2}$$

Therefore, we have $\Delta_2(D, \mathbf{d}) \geq \Delta_2(D-1, \mathbf{d})$. This shows that

$$c^{-\Delta_1(D,\mathbf{d})} + c^{-\Delta_2(D,\mathbf{d})} \leq c^{-\Delta_1(D-1,\mathbf{d})} + c^{-\Delta_2(D-1,\mathbf{d})} \leq 1.$$

When $f = D$, let $\mathbf{d}' := \{d_1, \ldots, d_{f-1}\}$. We have

$$\Delta_1(D, \mathbf{d}) = \Delta_1(D-1, \mathbf{d}') + \beta_{d_f} - \beta_{d_f - 1} \geq \Delta_1(D-1, \mathbf{d}')$$

and

$$\begin{aligned}\Delta_2(D, \mathbf{d}) &= \Delta_2(D-1, \mathbf{d}') - 2\frac{\alpha}{D} + 2\frac{\alpha}{D-1} + \beta_{d_f} - \beta_{d'} + \beta_{d'-d_f+2} \\ &\geq \Delta_2(D-1, \mathbf{d}') + \beta_{d_f} - \beta_{d'} + \beta_{d'-d_f+2},\end{aligned}$$

where $d' = D + \sum_{i=1}^{f}(d_i - 2)$. When $d' > 31$, we have $\beta_{d_f} - \beta_{d'} + \beta_{d'-d_f+2} \geq \beta_{d_f} - \beta_{31} + \beta_{31-d_f+2}$, and hence such a case can be reduced to the case when $d' = 31$. Because we can verify that our parameters satisfy

$$\beta_{d_f} - \beta_{d'} + \beta_{d'-d_f+2} \geq 0 \ \ (\forall (d', d_f) \text{ with } 3 \leq d_f < d' \leq 31), \tag{3}$$

we have $\Delta_2(D, \mathbf{d}) \geq \Delta_2(D-1, \mathbf{d}')$. This shows that

$$c^{-\Delta_1(D,\mathbf{d})} + c^{-\Delta_2(D,\mathbf{d})} \leq c^{-\Delta_1(D-1,\mathbf{d}')} + c^{-\Delta_2(D-1,\mathbf{d}')} \leq 1. \qquad \blacktriangleleft$$

▶ **Theorem 10.** *Algorithm 1 solves* FVS *in* $O^*(3.460^k)$ *time.*

4 Conclusion

In this paper, we give the theoretical analysis to the empirically fast branching algorithm for FVS. The proved running time is the current fastest among deterministic algorithms. We conclude the paper by giving two open problems.

First, we do not think that our analysis is tight. Can we significantly improve the analysis? Especially, we are interested in whether we can achieve $O^*(4^k)$ time without using the reduction to the linear matroid parity to solve subcubic instances. When allowing randomization, a simple random sampling algorithm runs in $O^*(4^k)$ time [1]. This random sampling algorithm also uses a degree-based argument but does not use the subcubic solver. Without the subcubic solver (i.e., $D \geq 3$ instead of $D \geq 4$), our analysis gives only $O^*(4.59^k)$. If this can be improved to $O^*(4^k)$, using the subcubic solver will further improve the running time.

Second, the proved running time is still far from actual running time against real-world instances. In this paper, we used the standard parameter of the solution size. Can we prove that the algorithm runs in FPT time for some parameter that is smaller in real-world instances. For example, Vertex Cover and Multiway Cut are known to be FPT parameterized by the gap between the solution size and LP lower bounds [8, 21].

References

1 Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized Algorithms for the Loop Cutset Problem. *J. Artif. Intell. Res.*, 12:219–234, 2000. doi:10.1613/jair.638.

2 Hans L. Bodlaender. On Disjoint Cycles. *Int. J. Found. Comput. Sci.*, 5(1):59–68, 1994. doi:10.1142/S0129054194000049.

3 Kevin Burrage, Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, Shev Mac, and Frances A. Rosamond. The Undirected Feedback Vertex Set Problem Has a Poly(k) Kernel. In *IWPEC 2006*, pages 192–202, 2006. doi:10.1007/11847250_18.

4 Yixin Cao. A Naive Algorithm for Feedback Vertex Set. In *SOSA 2018*, pages 1:1–1:9, 2018. doi:10.4230/OASIcs.SOSA.2018.1.

5 Yixin Cao, Jianer Chen, and Yang Liu. On Feedback Vertex Set: New Measure and New Structures. *Algorithmica*, 73(1):63–86, 2015. doi:10.1007/s00453-014-9904-6.

6 Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008. doi:10.1016/j.jcss.2008.05.002.

7 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *FOCS 2011*, pages 150–159, 2011. doi:10.1109/FOCS.2011.23.

8 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *TOCT*, 5(1):3:1–3:11, 2013. doi:10.1145/2462896.2462899.

9 Holger Dell, Thore Husfeldt, Bart M. P. Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The First Parameterized Algorithms and Computational Experiments Challenge. In *IPEC 2016*, pages 30:1–30:9, 2016. doi:10.4230/LIPIcs.IPEC.2016.30.

10 Rodney G. Downey and Michael R. Fellows. Fixed Parameter Tractability and Completeness. In *Complexity Theory: Current Research*, pages 191–225, 1992.

11 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

12 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56(5):25:1–25:32, 2009. doi:10.1145/1552285.1552286.

13 Serge Gaspers. *Exponential Time Algorithms - Structures, Measures, and Bounds*. VDM, 2010.

14 Kensuke Imanishi and Yoichi Iwata. Feedback Vertex Set solver, 2016. Entry to PACE challenge 2016. URL: http://github.com/wata-orz/fvs.

15 Yoichi Iwata. Linear-Time Kernelization for Feedback Vertex Set. In *ICALP 2017*, pages 68:1–68:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.68.

16 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching and FPT Algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016. doi:10.1137/140962838.

17 Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 0/1/All CSPs, Half-Integral A-Path Packing, and Linear-Time FPT Algorithms. In *FOCS 2018*, pages 462–473, 2018. doi:10.1109/FOCS.2018.00051.

18 Richard M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations 1972*, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9.

19 Krzysztof Kiljan and Marcin Pilipczuk. Experimental Evaluation of Parameterized Algorithms for Feedback Vertex Set. In *SEA 2018*, pages 12:1–12:12, 2018. doi:10.4230/LIPIcs.SEA.2018.12.

20 Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic Feedback Vertex Set. *Inf. Process. Lett.*, 114(10):556–560, 2014. doi:10.1016/j.ipl.2014.05.001.

21 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster Parameterized Algorithms Using Linear Programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.