A New Linear Kernel for Undirected Planar Feedback Vertex Set: Smaller and Simpler

Mingyu Xiao*

School of Computer Science and Engineering,
University of Electronic Science and Technology of China, China
myxiao@gmail.com

Abstract. We show that any instance I of the Feedback Vertex Set problem in undirected planar graphs can be reduced to an equivalent instance I' such that (i) the size of the instance and the size of the minimum feedback vertex set do not increase, (ii) and the size of the minimum feedback vertex set in I' is at least $\frac{1}{29}$ of the number of vertices in I'. This implies a $29k$ kernel for this problem with parameter k being the size of the feedback vertex set. Our result improves the previous results of $97k$ and $112k$.

Keywords: Kernelization, Feedback Vertex Set, Planar Graphs.

1 Introduction

A *feedback vertex set* of a directed or undirected graph is a subset of vertices intersecting all cycles in the graph. The Feedback Vertex Set problem asks us to find a feedback vertex set of minimum size in a given directed or undirected graph. This problem has applications in operating system, computer architecture communities, database system, rank aggregation and so on [18,16].

It is known that Feedback Vertex Set is NP-hard even in undirected planar graphs [12]. Due to the importance of this problem, Feedback Vertex Set has been extensively studied in exact and parameterized algorithms. Exact algorithms with running time better than the trivial bound of 2^n have been developed recently for both of Feedback Vertex Set in undirected graphs (UFVS) and Feedback Vertex Set in directed graphs (DFVS) [10,14,15], where n stands for the number of vertices in the graph. The running time bound for UFVS has been improved to $1.7266^n n^{O(1)}$ [18] but the best result for DFVS is still $1.9977^n n^{O(1)}$ by Razgon [15]. As for parameterized algorithms, we consider the parameterized problems with parameter k being the size of the feedback vertex set. Let k-UFVS (resp. k-DFVS) denote the problem of checking whether or not a given undirected (resp. directed) graph has a feedback vertex set of size at most k. A parameterized problem is *fixed-parameter tractable* (FPT) if it can be solved in $f(k)poly(n)$ time, where $poly(n)$ is an arbitrary polynomial function

* Supported by NFSC of China under the Grant 61370071 and Fundamental Research Funds for the Central Universities under the Grant ZYGX2012J069.

Q. Gu, P. Hell, and B. Yang (Eds.): AAIM 2014, LNCS 8546, pp. 288–298, 2014.

and $f(k)$ is an arbitrary computable function. There is a long list of contributions to fast FPT algorithms for k-UFVS [13,9,7,6]. Now it can be solved in $3.83^k n^{O(1)}$ time [6]. Whether k-DFVS is FPT or not had been a big open problem in parameterized algorithms for many years. Finally it was solved affirmatively by Chen *et. al.* [8].

Kernelization is one of the most active topics in parameterized algorithms. In this area, we aim to find a polynomial-time algorithm that reduces any instance (I, k) of a parameterized problem to an instance (I', k') of this problem such that $k' \leq k$ and the size of I' is bounded by a computable function $g(k')$ of k', where (I', k') is called a *kernel* of this problem and $g(k')$ is called the *size* of the kernel. We are interested in finding kernels of polynomial size. Whether or not k-DFVS has a polynomial kernel is still an open problem. As for k-UFVS, there is a long list of contributions to kernelizations. The first polynomial kernel for k-UFVS in general graphs was developed in [5], which was improved to a cubic kernel [4]. The current best result is the $4k^2$ kernel by Thomassé [17]. Linear kernels have been obtained for k-UFVS in some graph classes, such as bounded-genus graphs and H-minor free graphs [2,11]. For k-UFVS in planar graphs, Bodlaender and Pennikx [3] gave the first linear kernel of size $112k$, which was improved to $97k$ by Abu-Khzam and Khuzam [1]. In this paper, we gave a kernel of size $29k$, greatly improving previous results. In fact, all reduction rules in our algorithm are parameter-independent. It means that even if the parameter k is not part of the input, our algorithm can still reduce an instance I to an equivalent instance I' such that the size of I' is at most 29 times of the size of the solution to I'.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with possible parallel edges and self-loops. The vertex set and edge set of a graph G' are denoted by $V(G')$ and $E(G')$ respectively. If there is at least one edge between two vertices, we say that the two vertices are *adjacent*. For two adjacent vertices, any one is a *neighbor* of the other one. For a vertex subset or a subgraph V', the set of vertices in $V \setminus V'$ (or $V \setminus V(V')$ for a subgraph V') adjacent to at least one vertex in V' is denoted by $N(V')$. Furthermore, we use $N_H(V')$ to denote the set $N(V') \cap H$ for a vertex subset H (or $N(V') \cap V(H)$ for a subgraph H). We may denote a singleton set $\{a\}$ by a. For a vertex v, the *degree* of v is defined to be $d(v) = |N(v)|$ and the number of edges incident on v is denoted by $e(v)$. Then $e(v) \geq d(v)$ since the graph may contain parallel edges. A degree-2 vertex is called a *strong degree-2 vertex* if there are parallel edges between it and any of the two neighbors of it. A vertex is called *non-trivial* if it is a strong degree-2 vertex or a vertex of degree ≥ 3. We use (A, B, E) to denote a bipartite graph with edges between two vertex sets A and B.

The subgraph induced by a vertex subset $V' \subseteq V$ is denoted by $G[V']$. A path $v_1 v_2 \cdots v_r$ in the graph is called an *induced path* if there is exactly one edge between vertices v_i and v_{i+1} for $i \in \{1, 2, \cdots, r-1\}$ and no edge between vertices v_{i_1} and v_{i_2} with $|i_1 - i_2| \geq 2$. Note that we do not allow parallel edges in induced paths.

A graph without any cycle is called a *forest*. A *tree* is a connected graph without any cycle. In a forest, a degree-1 vertex is called a *leaf-vertex*, a vertex of degree ≥ 2 is called an *inner-vertex*, and a vertex of degree ≥ 3 is called a *branch-vertex*. A path in a tree is called a *branch* if it is a maximal path not containing any branch-vertex. Then after deleting all branch-vertices from a tree, each component is a branch of the tree. A branch is called a *leaf-branch* if it contains at least one leaf-vertex as its endpoint and a branch is called an *inner-branch* if all of its vertices are degree-2 vertices in the tree.

Contracting a vertex subset or a subgraph V' means deleting V', introducing a new vertex v, for any vertex $u \in V \setminus V'$ (or $u \in V \setminus V(V')$ for a subgraph V') adding x edges between v and u if there are x edges between V' and u before deleting V', and adding a self-loop incident on v if there is a cycle in the induced graph $G[V']$ (or subgraph V'). *Contracting an edge* means contracting the two endpoints of it. In our algorithm, we assume that the initial graph is a connected graph, because if the graph has more than one component we can simply take each component as the input to solve it.

2.1 Some Properties of Planar Graphs

Here we give some properties of planar graphs. One of them will be used to get a bound of the vertex number in our analysis and one of them will be used to show that after executing each of our operations on a planar graph the resulting graph is still a planar graph.

The famous Euler's formula gives a relation among the number of vertices, the number of edges and the number of faces of a planar graph. Let f be the number of faces of a planar graph $G = (V, E)$. It holds that

$$|V| - |E| + f = 2.$$

This formula can be used to get some upper bound of the edge number in a planar graph. For a planar graph, each edge belongs to 2 faces and each face contains at least 3 edges. For a bipartite planar graph, each edge belongs to 2 faces and each face contains at least 4 edges (since the bipartite graph has no odd cycle). Then by Euler's formula, we can get

Proposition 1. *For any planar graph, it holds that*

$$|E| \leq 3|V| - 6 \tag{1}$$

and, for any bipartite planar graph, it holds that

$$|E| \leq 2|V| - 4. \tag{2}$$

It is also easy to observe the following.

Proposition 2. *After applying any one of the following operations on a planar graph, the resulting graph is still planar*

(i) *deleting a vertex or an edge,*
(ii) *contracting an edge,*
(iii) *adding an edge between two adjacent vertices or two neighbors of a degree-2 or degree-3 vertex, and*
(iv) *adding a degree-2 vertex adjacent to a pair of adjacent vertices.*

3 Reduction Rules

Since that all the reduction operations in the paper are parameter-independent, we do not include the parameter when we discuss our reduction operations. A *reduction rule* is a procedure that takes a planar graph G as the input and outputs a planar graph G' and a set $S_0 \subseteq V$ such that for any minimum feedback vertex set S' of G', the union $S_0 \cup S'$ is a minimum feedback vertex set of G. In what follows, when introducing a reduction rule, we assume that all previous reduction rules cannot be applied on the current instance anymore.

It is easy to observe the following reduction rules to deal with vertices having self-loops and some vertices of degree at most 2 in the graph (except strong degree-2 vertices).

Rule 1. *If there is a cycle contains only one vertex (a self-loop incident on the vertex), delete the vertex from the graph and put it to* S_0.

Rule 2. *For a degree-1 vertex* v,
(i) *delete* v *from the graph if there is only one edge between* v *and its unique neighbor, and*
(ii) *delete* v *and its unique neighbor* u *and put* u *to* S_0 *if there are parallel edges between* v *and* u.

Rule 3. *For a degree-2 vertex* v *with two neighbors* u_1 *and* u_2,
(i) *delete* v *and add an edge between* u_1 *and* u_2 *if there are only two edges incident on* v, *and*
(ii) *delete* v *and* u_{i_1} *and put* u_{i_1} *to* S_0 *if there are parallel edges between* v *and* u_{i_1} *but only one edge between* v *and* u_{i_2}, *where* $\{i_1, i_2\} = \{1, 2\}$.

Note that after applying the above reduction rules all the vertices in the graph are non-trivial vertices. This property will be used in our analysis. In order to get a kernel of the problem, we need to reduce some local structures where a large number of vertices are only adjacent to a few vertices. Next, we will design some rules for this kind of local structures.

Lemma 1. *Let* $G = (V, E)$ *be a graph such that all above reduction rules can not be applied anymore. If there is a vertex subset* $V' \subset V$ *such that* $|V'| \geq 3$, *the induced graph* $G[V']$ *has no cycle, and* $|N(V')| \leq 2$, *then there is a minimum feedback vertex set of* G *containing all vertices in* $N(V')$.

Proof. We know that $|N(V')| = 2$ (because if $|N(V')| = 1$ then either Rule 2 or Rule 3 can be applied on any leaf-vertex in $G[V']$). Let $N(V') = \{w_1, w_2\}$.

There is no degree-0 vertex in $G[V']$, any degree-1 vertex in $G[V']$ is adjacent to both of w_1 and w_2 in G, and any degree-2 vertex in $G[V']$ is adjacent to at least one of w_1 and w_2 in G.

If there are at least three degree-1 vertices a_1, a_2 and a_3 in $G[V']$, then any minimum feedback vertex set S of G contains at least two vertices in $\{a_1, a_2, a_3, w_1, w_2\}$. Note that after deleting w_1 and w_2 from G, no vertex in V' is contained in a cycle. We know that $\{w_1, w_2\}$ intersects any cycle containing at least one vertex in $V' \cup \{w_1, w_2\}$. Thus, $S' = S \setminus \{a_1, a_2, a_3, w_1, w_2\} \cup \{w_1, w_2\}$ is still a minimum feedback vertex set of G. If there are at most two degree-1 vertices in $G[V']$, then $G[V']$ can only be a path P. There are at least three vertices in P since $|V'| \geq 3$. Any no-endpoint vertex in P is adjacent to at least one vertex in $\{w_1, w_2\}$ in G. It is easy to see that we need at least two vertices to intersect all cycles in the subgraph $G[V' \cup \{w_1, w_2\}]$. Then any minimum feedback vertex set S of G contains at least two vertices in $V' \cup \{w_1, w_2\}$. If S does not contain both of w_1 and w_2, then we replace the vertices in $(V' \cup \{w_1, w_2\}) \cap S$ with $\{w_1, w_2\}$ in S to get another minimum feedback vertex set of G. So there is always a minimum feedback vertex set containing both vertices in $N(V')$. □

This lemma can be used to get reduction rules to deal with some vertex-cuts of size at most 2. However, our algorithm only uses a very special case where $|V'| = 3$.

Rule 4. *Let $P = v_1v_2v_3$ be an induced path in G such that $|N(P)| \leq 2$. Delete $N(P)$ from the graph and put all vertices in $N(P)$ to S_0. (See Figure 1 (a))*

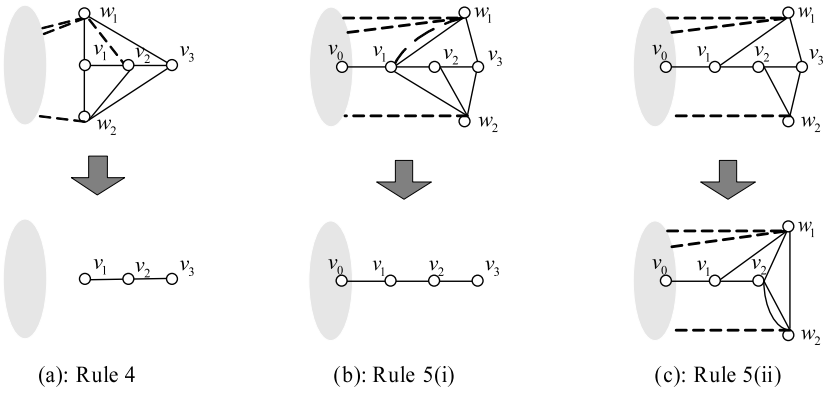


Fig. 1. Illustrations for Rules 4 and 5

The next three rules are also used to deal with some induced paths with small number of neighbors.

Rule 5. *Let $v_0v_1v_2v_3$ be an induced path in G where each vertex in $V' = \{v_1, v_2, v_3\}$ is a non-trivial vertex and $|N(V')| = 3$. Let $N(V') = \{v_0, w_1, w_2\}$ and $V'' = \{v_1, v_2, v_3, w_1, w_2\}$.*
(i) *If the size of the minimum feedback vertex set of the induced subgraph $G[V'']$ is at least 2, delete w_1 and w_2, and put the two vertices to S_0; and*
(ii) *If the size of the minimum feedback vertex set of the induced subgraph $G[V'']$ is 1, add a new edge between w_1 and w_2, and contract edge v_2v_3 into a single vertex v_2. (See Figure 1 for an illustration)*

Proof. Each vertex in V' is adjacent to at least one vertex in $\{w_1, w_2\}$ and v_3 is adjacent to both of w_1 and w_2, because any vertex in V' is non-trivial and none of v_2 and v_3 can be adjacent to v_0.

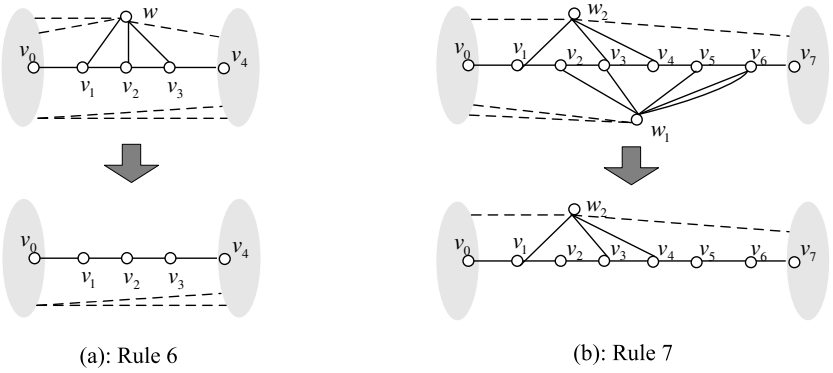
(i): When the size of the minimum feedback vertex set of $G[V'']$ is at least 2, any minimum feedback vertex set S of G contains at least two vertices in V''. However, w_1 and w_2 intersect all cycles containing some vertex in V''. Then $S \setminus V'' \cup \{w_1, w_2\}$ is still a minimum feedback vertex set of G. There is a minimum feedback vertex set containing w_1 and w_2 and then we can include them to the solution set directly.

(ii): When the size of the minimum feedback set of $G[V'']$ is 1, the vertex intersecting all cycles in $G[V'']$ can only be v_2 or v_3. For this case, there are no parallel edges between v_3 and a neighbor of it. Let G' be the resulting graph after executing the operation in (ii) on G and S' be a minimum feedback vertex set of G'. We show that S' is also a minimum feedback vertex set of G.

First, we show that S' is a feedback vertex set of G. Case 1. $v_2 \in S'$: Note there are no parallel edges incident on v_3 in G. The two graphs $G \setminus \{v_2\}$ and $G' \setminus \{v_2\}$ are almost the same except the degree-2 vertex v_3 in $G \setminus \{v_2\}$ is replaced with an edge in $G' \setminus \{v_2\}$. Then there is no cycle in $G \setminus S'$ if there is no cycle in $G \setminus S'$. Case 2. $v_2 \notin S'$: Now S' contains at least two vertices in $\{w_1, w_2, v_1\}$. Assume to the contrary that there is a cycle C in $G \setminus S'$. The cycle C must contain some edges in the induced subgraph $G[V'']$. Furthermore, C is contained in $G[V'']$, because if C also contains an edge not in $G[V'']$ then C should pass through at least two vertices in $\{w_1, w_2, v_1\}$ in $G \setminus S'$. Then the cycle C is either $v_2w_{i_0}$ or $v_2v_3w_{i_0}$, where $w_{i_0} \in \{w_1, w_2\}$. Both cases imply a cycle $v_2w_{i_0}$ in $G' \setminus S'$, a contradiction. So S' is a feedback vertex set of G.

To show that S' is also a minimum feedback vertex set of G, we only need to prove that the size of a minimum feedback vertex of G is not greater than the size of a minimum feedback vertex of G'. Let S be a minimum feedback vertex set in G. If S contains only one vertex v^* in V'', then v^* can only be v_2 or v_3. For this case, we can see that $S \setminus \{v^*\} \cup \{v_2\}$ is a feedback vertex set of G'. If S contains at least two vertices in V'', then $S' = S \setminus V'' \cup \{w_1, w_2\}$ is still a minimum feedback vertex set of G. Furthermore, S' is also a feedback vertex set of G'. Thus, the size of a minimum feedback vertex of G' will not be larger than the size of a minimum feedback vertex of G. □

The next two rules are firstly used in [1] to get a kernel for the problem. We also need to use them. Figure 2 gives illustrations for these two rules.



(a): Rule 6 (b): Rule 7

Fig. 2. Illustrations for Rules 6 and 7

Rule 6. *Let $v_0v_1v_2v_3v_4$ be an induced path in G where v_1, v_2 and v_3 are three degree-3 vertices having a common neighbor w. Delete w from the graph and put w to S_0.*

To prove the correctness of this reduction rule, we only need to show that there is at least one minimum feedback vertex set of G containing w. Let S be a minimum feedback vertex set of G not containing w. Since w is not in S, we know that at least two vertices v_{i_1} and v_{i_2} in $\{v_1, v_2, v_3\}$ are in S. It is easy to see that any cycle that contains at least one of v_{i_1} and v_{i_2} also contains at least one of w and v_1. Then $S' = S \setminus \{v_{i_1}, v_{i_2}\} \cup \{w, v_1\}$ is a minimum feedback vertex set of G containing w.

Rule 7. *Let $v_0v_1v_2v_3v_4v_5v_6v_7$ be an induced path in G where each vertex in $V' = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ is a vertex of degree ≥ 3 and $|N(V')| = 4$. Let $N(V') = \{v_0, v_7, w_1, w_2\}$. Assume that $|N_{V'}(w_1)| \geq |N_{V'}(w_2)|$. Delete w_1 from the graph and put w_1 to S_0.*

We can verify that there is a minimum feedback vertex set of G containing w_1 in the above rule. We omit the full proof here since it can be obtained in [1].

Our algorithm takes a planar graph as the input, iteratively execute the above reduction rules in order until none of them can be applied anymore, and then return the resulting planar graph. It is easy to verify that after executing each of our reduction rules the resulting graph is still planar by Proposition 2. The algorithm runs in polynomial time, since each reduction rule can be applied in polynomial time and reduces at least one vertex in the graph. We are only interested in the size of the resulting graph, so we omit a more detailed analysis of the complexity of the algorithm. In fact, all of our reduction rules can be applied in general graphs. We only analyze a kernel for planar graphs. We call a graph *reduced* if none of the above reduction rules can be applied.

4 The Analysis

In this section, we assume that G is a reduced planar graph. Let S be an arbitrary feedback vertex set of G, $k = |S|$ and $n = |V(G)|$. We will prove that $n \leq 29k - 57$.

Let F be the remaining graph after deleting the solution set S from the graph G, i.e., $F = G[V \setminus S]$. Then F is a forest. Note that each degree-1 vertex in F is adjacent to at least 2 vertices in S in G and each degree-2 vertex in F is adjacent to at least one vertex in S in G since all vertices in G are non-trivial. We analyze the number of vertices in the forest F. Assume that F has l leaf-vertices and c connected components.

Lemma 2. *It holds that*

$$l \leq 2k + 2c - 4. \tag{3}$$

Proof. For each component of F that is not a single edge, we contract all inner-vertices in it into a single inner-vertex. For each component of F that is a single edge e, we introduce a degree-2 vertex adjacent to the two endpoints of e and deleting e. Then the new added vertex will become an inner vertex in this component. Let F' be the resulting forest. Then F' has c inner-vertices. We consider the bipartite graph $H = (A = L, B = I \cup S, E)$, where L is the set of leaf-vertices in F' and I is the set of inner-vertices in F'. There is an edge between $a \in L$ and $b \in I$ in H if there is an edge between a and b in F'. There is an edge between $a \in L$ and $s \in S$ in H if there is an edge between a and s in G. The bipartite graph H is still a planar graph since it can be obtained from G by contracting some edges, deleting some edges or adding a degree-2 vertex adjacent to two endpoints of an edge. By (2), the number of edges in H is at most $2|V(H)| - 4 \leq 2(l + c + k) - 4$. On the other hand, each vertex $a \in L$ is adjacent to at least two vertices in S (since G has only no-trivial vertices and then each leaf-vertex in F is adjacent to at least two vertices in S) and adjacent to one inner-vertex in I. Then there are at least $3|L|$ edges between L and $I \cup S$. We get $3l \leq 2(l + c + k) - 4$, which is (3). □

Lemma 3. *The number of branch-vertices in F is at most $l - 2c$ and the number of inner-branches in F is at most $l - 3c$.*

Proof. Assume that the c trees in F have $l_1, l_2, \cdots, l_c$ leaf-vertices respectively. It is easy to see that a tree with l_i leaf-vertices has at most $l_i - 2$ branch-vertices and at most $l_i - 3$ inner-branches. Then F has at most $\sum_{i=1}^{c}(l_i - 2) = l - 2c$ branch-vertices and at most $\sum_{i=1}^{c}(l_i - 3) = l - 3c$ inner-branches. □

A sub-path P of a branch is *good* if $|N_S(P)| \geq 3$, i.e., a good sub-path is adjacent to at least 3 vertices in S. We use the following method to partition each branch of F into several sub-paths, called *chains*, such that each vertex in the branch is contained in exactly one of the sub-paths. Let $Q = v_1 v_2 \cdots v_q$ be a branch, where we assume that v_q is a leaf-vertex in F if Q is a leaf-branch.

If Q is not a good path, we take Q as a single chain. Otherwise we partition Q into chains $P_1 = v_1v_2\cdots v_{i_1}, P_2 = v_{i_1+1}v_{i_1+2}\cdots v_{i_2}, \cdots, P_r = v_{i_{r-1}+1}v_{i_{r-1}+2}\cdots v_q$ such that for any chain $P_j = v_{i_{j-1}+1}v_{i_{j-1}+2}\cdots v_{i_j}$, either (i) P_j is good and $v_{i_{j-1}+1}v_{i_{j-1}+2}\cdots v_{i_j-1}$ is not good or (ii) P_j is not good and it holds $i_j = q$.

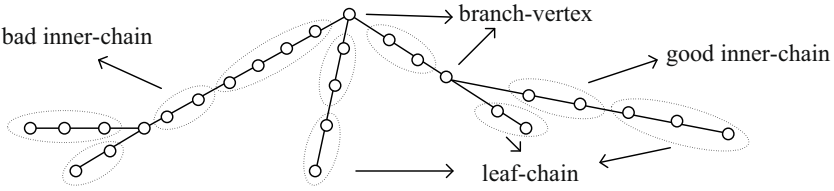


Fig. 3. Partitioning each branch in a tree into chains

We fix such a partition $\mathcal{P}$ of F and analyze the number of vertices in it. See Figure 3 for an illustration for the partition. We can see that each vertex of degree ≤ 2 in F is contained in a chain. A chain is *good* if it is a good path and *bad* otherwise. A chain is called a *leaf-chain* if it contains at least one leaf-vertex in F and an *inner-chain* if it does not contain any leaf-vertex in F. We will analyze the size and number of the chains. According to the way we partition a branch into chains, we know that

Lemma 4. *The number of bad inner-chains in $\mathcal{P}$ is at most $l-3c$ and each bad inner-chain contains at most 5 vertices.*

Proof. In the partition $\mathcal{P}$ of F, each inner-branch can contain at most one bad inner-chain and each leaf-branch can not contain any bad inner-chain. Then we know that the number of bad inner-chains in $\mathcal{P}$ is bounded by the number of inner-branches in F. By Lemma 3, the number of inner-branches in F is at most $l-3c$, which implies the first claim in the lemma. Next, we consider the second claim. Assume to the contrary that a bad inner-chain P contains at least 6 vertices. Since G is a reduced graph and all vertices in it are non-trivial, we know that each vertex in the bad inner-chain P is adjacent to at least one vertex in S. Then the condition of either Rule 6 or Rule 7 will hold, which implies a contradiction. □

Lemma 5. *The number of bad leaf-chains in $\mathcal{P}$ is at most $2k+2c-4$ and each bad leaf-chain contains at most 2 vertices.*

Proof. Since each leaf-chain contains at least one leaf-vertex, we know that the number of bad leaf-chains in $\mathcal{P}$ is bounded by the number of leaf-vertices in F. By Lemma 2, the number of leaves in F is at most $2k+2c-4$. We get the first claim. For the second claim, we can see that if a bad leaf-chain contains at least 3 vertices, then either Rule 4 or Rule 5 can be applied. □

Lemma 6. *The number of good chains in $\mathcal{P}$ is at most $2k-4$ and each good chain contains at most 6 vertices.*

Proof. We consider a bipartite planar graph $H = (A, B, E)$, where the number of vertices in A equals to the number of good chains in $\mathcal{P}$, each vertex $a \in A$ is corresponding a good chain Q_a in $\mathcal{P}$, and $B = S$. The edge set E is defined by this: there is an edge between vertices $a \in A$ and $s \in B$ if and only if there is an edge between s and a vertex in Q_a. Note that H can be obtained from G by contracting all edges in good chains and then deleting some vertices and edges. By Proposition 2, we know that H is still a planar graph. According to the definition of good chains, we know that each vertex $a \in A$ is adjacent to at least 3 vertices in B. Then there are at least $3|A|$ edges between A and B. By (2), we have that $3|A| \leq 2(|A| + |B|) - 4 = 2|A| + 2k - 4$. Then we get

$$|A| \leq 2k - 4.$$

For the second claim, we first assume to the contrary that a good chain Q contains at least 7 vertices. We look at the sub-path Q' containing only the first 6 vertices in Q. According to the definition of chains, we know that Q' should be a bad path. Then either Rule 6 or Rule 7 can be applied on Q', a contradiction to the fact that the graph is a reduced graph. □

Lemma 7. *The number of vertices in F is at most $28k - 57$.*

Proof. Let n_2 denote the number of degree-1 and degree-2 vertices in F and n_3 denote the number of vertices of degree ≥ 3, which are branch-vertices in F. Then $n_3 \leq l - 2c$ by Lemma 3. Each vertex of degree ≤ 2 is contained in a chain in $\mathcal{P}$. Each chain is either a good chain or a bad chain. Each bad chain is either a bad inner-chain or a bad leaf-chains. By Lemma 4, Lemma 5 and Lemma 6, we get

$$\begin{aligned}
|V(F)| &= n_2 + n_3 \\
&\leq n_2 + (l - 2c) \\
&\leq 5(l - 3c) + 2(2k + 2c - 4) + 6(2k - 4) + (l - 2c) \\
&= 6l + 16k - 13c - 32 \\
&\leq 6(2k + 2c - 4) + 16k - 13c - 32 \qquad \text{by (3)} \\
&= 28k - c - 56 \\
&\leq 28k - 57.
\end{aligned}$$

□

The number of vertices in G is $|V(F)| + |S| \leq 28k - 57 + k = 29k - 57$. Therefore, we obtain the following theorem

Theorem 1. *Let G be a planar graph such that none of the above reduction rules can be applied on it. Then the size of a minimum feedback vertex set of G is more than $\frac{|V(G)|}{29}$.*

References

1. Abu-Khzam, F.N., Bou Khuzam, M.: An improved kernel for the undirected planar feedback vertex set problem. In: Thilikos, D.M., Woeginger, G.J. (eds.) IPEC 2012. LNCS, vol. 7535, pp. 264–273. Springer, Heidelberg (2012)
2. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) kernelization. In: FOCS 2009, pp. 629–638. IEEE Computer Society, Washington, DC (2009)
3. Bodlaender, H.L., Penninkx, E.: A Linear Kernel for Planar Feedback Vertex Set. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 160–171. Springer, Heidelberg (2008)
4. Bodlaender, H.L., van Dijk, T.C.: A Cubic Kernel for Feedback Vertex Set and Loop Cutset. Theory Comput. Syst. 46(3), 566–597 (2010)
5. Burrage, K., Estivill-Castro, V., Fellows, M.R., Langston, M.A., Mac, S., Rosamond, F.A.: The undirected feedback vertex set problem has a poly(k) kernel. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 192–202. Springer, Heidelberg (2006)
6. Cao, Y., Chen, J., Liu, Y.: On feedback vertex set new measure and new structures. In: Kaplan, H. (ed.) SWAT 2010. LNCS, vol. 6139, pp. 93–104. Springer, Heidelberg (2010)
7. Chen, J., Fomin, F., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. J. Comput. Syst. Sci. 74, 1188–1198 (2008)
8. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. J. ACM 55, 1–19 (2008)
9. Dehne, F., Fellows, M.R., Langston, M.A., Rosamond, F.A., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 859–869. Springer, Heidelberg (2005)
10. Fomin, F.V., Gaspers, S., Pyatkin, A.V., Razgon, I.: On the minimum feedback vertex set problem: Exact and enumeration algorithms. Algorithmica 52(2), 293–307 (2008)
11. Fomin, F.V., Lokshtanov, D., Saurabh, S., Thilikos, D.M.: Bidimensionality and kernels. In: SODA 2010, Philadelphia, PA, USA, pp. 503–510 (2010)
12. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco (1979)
13. Guo, J., Gramm, J., Huffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. J. Comput. Syst. Sci. 72(8), 1386–1396 (2006)
14. Razgon, I.: Exact computation of maximum induced forest. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 160–171. Springer, Heidelberg (2006)
15. Razgon, I.: Computing minimum directed feedback vertex set in $O(1.9977^n)$. In: 10th Italian Conference on Theoretical Computer Science, ICTCS 2007, Rome, Italy, pp. 70–81 (2007)
16. Silberschatz, A., Galvin, P.: Operating System Concepts, 4th edn. Addison-Wesley (1994)
17. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM Transactions on Algorithms 6(2) (2010)
18. Xiao, M., Nagamochi, H.: An Improved Exact Algorithm for Undirected Feedback Vertex Set. Journal of Combinatorial Optimization (2014), doi: 10.1007/s10878-014-9737-x; A preliminary version appears as: Xiao, M., Nagamochi, H.: An Improved Exact Algorithm for Undirected Feedback Vertex Set. In: Widmayer, P., Xu, Y., Zhu, B. (eds.) COCOA 2013. LNCS, vol. 8287, pp. 153–164. Springer, Heidelberg (2013)