Kernelization for Counting Problems on Graphs: Preserving the Number of Minimum Solutions

(article starts on next page)

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 29, no. 1, pp. 1–28 (2025)
DOI: 10.7155/jgaa.v29i1.2963

Kernelization for Counting Problems on Graphs: Preserving the Number of Minimum Solutions

Bart M. P. Jansen[1] *Bart van der Steenhoven*[2]

[1]Eindhoven University of Technology, The Netherlands
[2]Chalmers University of Technology and University of Gothenburg, Sweden

Submitted: August 2024 Accepted: February 2025 Published: March 2025

Article type: Regular Paper Communicated by: Meirav Zehavi

Abstract. A kernelization for a parameterized decision problem $\mathcal{Q}$ is a polynomial-time preprocessing algorithm that reduces any parameterized instance (x, k) into an instance (x', k') whose size is bounded by a function of k alone and which has the same YES/NO answer for $\mathcal{Q}$. Such preprocessing algorithms cannot exist in the context of counting problems, when the answer to be preserved is the number of solutions, since this number can be arbitrarily large compared to k. However, we show that for counting minimum feedback vertex sets of size at most k, and for counting minimum dominating sets of size at most k in a planar graph, there is a polynomial-time algorithm that either outputs the answer or reduces to an instance (G', k') of size polynomial in k with the same number of minimum solutions. This shows that a meaningful theory of kernelization for counting problems is possible and opens the door for future developments. Our algorithms exploit that if the number of solutions exceeds $2^{\text{poly}(k)}$, the size of the input is exponential in terms of k so that the running time of a parameterized counting algorithm can be bounded by poly(n). Otherwise, we can use gadgets that slightly increase k to represent choices among $2^{\mathcal{O}(k)}$ options by only poly(k) vertices.

Bart M. P. Jansen: Funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803421, ReduceSearch). Bart van der Steenhoven: Supported by Project No. ICT22-029 of the Vienna Science Foundation (WWTF). A preliminary version of this paper appeared in the proceedings of the 18th International Symposium on Parameterized and Exact Computation, IPEC 2023 [20].

E-mail addresses: b.m.p.jansen@tue.nl (Bart M. P. Jansen) bartj@chalmers.se (Bart van der Steenhoven)

1 Introduction

Background and Motivation. Counting problems, whose answer is an integer giving the number of objects of a certain kind rather than merely YES or NO, have important applications in fields of research such as artificial intelligence [28, 29], statistical physics [21, 24, 37] and network science [26]. They have been studied extensively in classical complexity, underpinning fundamental results such as Toda's theorem [34] and the #P-completeness of the permanent [35]. A substantial research effort has targeted the parameterized complexity of counting problems, leading to parametric complexity-notions like #W[1]-hardness [10, 14, 25] and fixed-parameter tractable (FPT) algorithms to solve several counting problems. For example, FPT algorithms were developed to count the number of size-k vertex covers [13], or the number of occurrences of a size-k pattern graph H in a host graph G [11].

This paper is concerned with an aspect of parameterized algorithms which has been largely neglected for counting problems: that of efficient preprocessing with performance guarantees, i.e., kernelization. A kernelization for a parameterized decision problem $\mathcal{Q}$ is a polynomial-time preprocessing algorithm that reduces any parameterized instance (x, k) into an instance (x', k') whose size is bounded by a function of k alone and which has the same YES/NO answer for $\mathcal{Q}$. Over the last decade, kernelization has developed into an important subfield of parameterized algorithms, as documented in a textbook dedicated to the topic [15]. Given the success of kernelization for decision problems, one may wonder: can a theory of provably-efficient preprocessing for counting problems be developed?

Consider a prototypical problem such as FEEDBACK VERTEX SET in undirected graphs, in which the goal is to find a small vertex set whose removal breaks all cycles. What could be an appropriate notion of counting kernelization for such a problem? The concept of efficient preprocessing towards a provably small instance with the same answer could be instantiated as follows: given a pair (G, k), the preprocessing algorithm should output a pair (G', k') whose size is bounded by a function of k such that the number of size-k feedback vertex sets in G is equal to the number of size-k' feedback vertex sets in G'. However, this task is clearly impossible. Given a graph consisting of a length-n cycle with parameter $k = 1$, the number of solutions is n which can be arbitrarily large compared to k, while for any reduced instance (G', k') of size bounded in k, the number of solutions can be at most $2^{|V(G')|} \leq f(k)$. Without allowing the size of the reduced instance to depend on n, it seems that preprocessing while preserving the answer to the counting problem is impossible.

Over the years, there have been two approaches to deal with this obstacle[1]. Thurley [33], inspired by concepts in earlier work [27], proposed a notion of counting kernelization which effectively reduces a counting problem to an enumeration problem. He considered problems such as VERTEX COVER and d-HITTING SET. In his framework, the preprocessing algorithm has to output an instance of size bounded by a function of k, in such a way that for any solution to the reduced instance, we can efficiently determine to how many solutions of the original instance it corresponds. Hence by enumerating *all* solutions on the reduced instance, we can obtain the number of solutions to the original instance. A significant drawback of this approach therefore lies in the fact that to solve the counting problem on the original instance, we have to enumerate all solutions on the reduced instance. Since counting can potentially be done much faster than enumeration, it is not clear that this preprocessing step is always beneficial.

[1]A third [23] approach was published after the extended abstract of our manuscript appeared; it allows a polynomial-time lifting step to compute the number of solutions to the original instance from the number of solutions to the reduced instance. We discuss it in the section on related work below.

A second notion for counting kernelization was proposed by Kim, Serna, and Thilikos [22, 31]. Their framework (which also applies to FEEDBACK VERTEX SET) considers two types of algorithms: a *condenser* that maps an input instance (G, k) to an instance (G', k') of an auxiliary *annotated* problem involving weights on the vertices of G', and an *extractor* that recovers (typically not in polynomial time) the number of solutions to (G, k) from the weighted instance (G', k'). The number of vertices of G' is required to be bounded in k, but the weights are allowed to be arbitrarily large, thereby sidestepping the issue described above. This means that in terms of the total encoding size, the weighted graph (G', k') is not guaranteed to be smaller than (G, k) and in general the total number of bits needed to encode the weighted graph cannot be bounded by a function of k alone. The condenser-extractor framework has the same drawback as the framework by Thurley: a standard counting problem is reduced to a more complicated type of problem, in this case one involving weights and annotations.

The goal of this paper is to show that there is an alternative way to overcome the obstacle for counting kernelization, which leads to a notion of preprocessing in which the problem to be solved on the reduced instance is of exactly the same nature as the original. Our solution is inspired by the typical behavior of kernelization algorithms for decision problems: we formalize the option of already finding the answer during the preprocessing phase. Note that many algorithms, such as the famous Buss [8] kernelization for VERTEX COVER, work by applying reduction rules to arrive at the reduced instance, or discover the YES/NO answer to the decision problem during preprocessing. Our kernelization algorithms for counting problems will have the same behavior: they will either reduce to a poly(k)-sized instance of the same problem that has exactly the same answer to the counting problem, or they outright answer the counting problem during their polynomial-time computation. To our initial surprise, such preprocessing algorithms exist for several classic problems.

Our Results. To begin the exploration of this new type of counting kernelization, we revisit two prominent graph problems: FEEDBACK VERTEX SET in general undirected graphs and DOMINATING SET in planar graphs. The decision versions of these problems (does graph G have a solution of size at most k?) have kernels with $\mathcal{O}(k^2)$ [19, 32] and $\mathcal{O}(k)$ vertices [1, 7, 17], respectively. We consider the problem of counting the number of *minimum-size* solutions, parameterized by the size k of a minimum solution. (We discuss counting inclusion-minimal solutions in the conclusion.) For a graph G and integer k, we denote by $\#\mathrm{minFVS}(G, k)$ the number of minimum feedback vertex sets in G of size at most k in G. Hence $\#\mathrm{minFVS}(G, k)$ is equal to 0 if the feedback vertex number of G exceeds k, and otherwise is equal to the number of minimum solutions. The analogous concept for minimum dominating sets is denoted $\#\mathrm{minDS}(G, k)$. Our result for FEEDBACK VERTEX SET reads as follows.

Theorem 1. *There is a polynomial-time algorithm that, given a graph G and integer k, either*

- *outputs $\#\mathrm{minFVS}(G, k)$, or*
- *outputs a graph G' and integer k' such that $\#\mathrm{minFVS}(G, k) = \#\mathrm{minFVS}(G', k')$ and $|V(G')| = \mathcal{O}(k^5)$ and $k' = \mathcal{O}(k^5)$.*

For DOMINATING SET on planar graphs, we give an analogous algorithm that either outputs $\#\mathrm{minDS}(G, k)$ or reduces to a *planar* instance (G', k') with $|V(G')|, k' = \mathcal{O}(k^3)$ such that $\#\mathrm{minDS}(G, k) = \#\mathrm{minDS}(G', k')$. Hence if the parameter is small, the task of counting the number of minimum solutions can efficiently be reduced to the *same* task on a provably small instance.

The high-level approach is the same for both problems. We use insights from existing kernels for the decision version of the problem to reduce an input instance (G, k) into one (G', k') with the same number of minimum solutions, such that G' can be decomposed into a "small" core together with poly(k) "simply structured but potentially large" parts. For Dominating Set, this takes the form of a protrusion decomposition; for Feedback Vertex Set the decomposition is more elementary. Then we consider two cases. If $|V(G)| > 2^k$, we employ an FPT algorithm running in time $2^{\mathcal{O}(k)} \cdot \text{poly}(n)$ to count the number of minimum solutions and output it. Since $n > 2^k$, this step runs in polynomial time. If $|V(G)| \leq 2^k$, then we show that each of the poly(k) simply structured parts can be replaced with a gadget of size poly(k) without affecting the number of minimum solutions. In this step, we typically increase the size of minimum solutions slightly to allow a small vertex set to encode exponentially many potential solutions. For example, an instance of Feedback Vertex Set consisting of a cycle of length 2^{10} (which has 2^{10} different optimal solutions), can be reduced to the graph consisting of 10 pairs (a_i, b_i), each pair connected by two parallel edges. The latter graph also has 2^{10} minimum solutions, each of size 10. To carry out this approach, the most technical part is to show how to decompose the input instance into parts in which it is easy to analyze how many different choices an optimal solution can make.

Related Work. Having described our main results, we compare them to the independent work on preprocessing for counting problems by Lokshtanov, Misra, Saurabh, and Zehavi [23] that appeared shortly after publication of the extended abstract of this work. They propose a framework in which a preprocessing step consists of two polynomial-time algorithms: an algorithm reduce that, given a parameterized input (x, k) to a counting problem, outputs a reduced instance (x', k') whose size is bounded in terms of k; and an algorithm lift that, given (x, k) and the number of solutions to (x', k'), outputs the number of solutions to (x, k). In this framework, they developed a preprocessing step for counting the number of vertex covers of size at most k in a given graph G. They present a polynomial-time reduce algorithm that maps any instance (G, k) to an instance (G', k') on $\mathcal{O}(k^6)$ vertices, along with a polynomial-time algorithm that can recover the number of size-($\leq k$) vertex covers of G from the number of size-($\leq k'$) vertex covers of G'. They therefore manage to obtain a preprocessing step suitable for counting solutions that are not minimum or minimal, at the cost of requiring a polynomial-time post-computation on the answer to the reduced instance.

In addition to the algorithm for counting size-k vertex covers, Lokshtanov, Misra, Saurabh, and Zehavi [23] also present results for the Planar $\mathcal{F}$-Deletion problem. For a fixed finite set of graphs $\mathcal{F}$, which is required to contain at least one planar graph, a solution to this problem on a graph G consists of a size-($\leq k$) vertex set S such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor. For this broad class of problems, which includes Feedback Vertex Set, they give a slightly weaker form of preprocessing algorithm. Here the reduce algorithm outputs an instance to a *different* and annotated counting problem, of total encoding size $k^{\mathcal{O}(1)}$. The lift algorithm then computes the number of size-($\leq k$) solutions to Planar $\mathcal{F}$-Deletion on the original graph from the number of solutions to the annotated problem on the reduced graph. This preprocessing scheme therefore has the benefit of applying to a large class of problems, but the downside of reducing to a more difficult counting task.

Apart from the mentioned positive results, Lokshtanov, Misra, Saurabh, and Zehavi [23] also present *lower-bound* techniques in their paper that allow them to rule out the impossibility of obtaining preprocessing schemes for which the size of reduced instances is polynomial in k, subject to complexity-theoretic conjectures.

Organization. The remainder of the paper is structured as follows. After presenting preliminaries in Section 2, we illustrate our approach for Feedback Vertex Set in Section 3. The more technical application to Dominating Set on planar graphs is given in Section 4. We conclude in

Section 5 with a reflection on the potential of this approach to counting kernelization.

2 Preliminaries

2.1 Graphs

All graphs we consider are undirected; they may have parallel edges but no self-loops. A graph G therefore consists of a set $V(G)$ of vertices and a multiset $E(G)$ of edges of the form $\{u, v\}$ for distinct $u, v \in V(G)$. For a vertex $v \in V(G)$, we refer to the *open neighborhood* of v in G as $N_G(v)$ and to the *closed neighborhood* of v as $N_G[v]$. For a set of vertices $X \subseteq V(G)$, the open and closed neighborhoods are defined as $N_G(X) = (\bigcup_{v \in X} N_G(v)) \setminus X$ and $N_G[X] = \bigcup_{v \in X} N_G[v]$. The degree of vertex v in graph G, denoted by $\deg_G(v)$, is equal to the number of edges incident to v in G. We refer to the subgraph of G induced by a vertex set $X \subseteq V(G)$ as $G[X]$. We use $G - X$ as a way to write $G[V(G) \setminus X]$ and $G - v$ as a shorthand for $G - \{v\}$. A graph H is a *minor* of G if H can be formed by contracting edges of a subgraph of G.

A graph is *planar* if it can be embedded in the plane in such a way that its edges intersect only at their endpoints. Such an embedding is called a *planar embedding* of G. We make use of the following two properties of planar graphs.

Theorem 2 (Wagner's theorem)**.** *A graph G is planar if and only if G contains neither K_5 nor $K_{3,3}$ as a minor.*

Lemma 1 ([15, Lemma 13.3])**.** *Let G be a planar graph, $C \subseteq V(G)$, and let N_3 be a set of vertices from $V(G) \setminus C$ such that every vertex from N_3 has at least three neighbors in C. Then, $|N_3| \leq \max\{0, 2|C| - 4\}$.*

A *feedback vertex set* of a graph G is a set $S \subseteq V(G)$ such that $G - S$ is a forest, i.e., acyclic. The *feedback vertex number* of a graph is the size of a smallest feedback vertex set of that graph. A *dominating set* of a graph G is a set $D \subseteq V(G)$ such that $N_G[D] = V(G)$. The *domination number* of a graph is the size of a smallest dominating set of that graph. We say that a set $X \subseteq V(G)$ dominates $U \subseteq V(G)$ if $U \subseteq N_G[X]$.

We define $V_{\neq 2}(G)$ to be the set of vertices of graph G that do not have degree two. We refer to a *chain* C of G as a connected component of $G - V_{\neq 2}(G)$. We say that a chain C is a *proper chain* if $N_G(C) \neq \emptyset$ and we then refer to $N_G(C)$ as the *endpoints* of C.

2.2 Treewidth

In order to prove the main theorem of Section 4, we will need to equip ourselves with one additional tool, which we describe next. For completeness, we will first give the definition of a tree decomposition, which is needed to formally define the notion of a treewidth modulator below.

Definition 3 (Tree decomposition)**.** A tree decomposition of a graph G is a pair (T, χ), where T is a rooted tree and χ is a function that assigns a bag $\chi(w) \subseteq V(G)$ to every node $w \in V(T)$ such that

- $\bigcup_{w \in V(T)} \chi(w) = V(G)$,
- for all $\{u, v\} \in E(G)$, there exists a node $w \in V(T)$ such that $\{u, v\} \subseteq \chi(w)$, and
- for every $v \in V(G)$, the set $\{w \in V(T) \mid v \in \chi(w)\}$ induces a non-empty connected subtree of T.

The *width* of a tree decomposition is $\max_{w \in V(T)} |\chi(w)| - 1$. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum possible width of a tree decomposition of G.

Definition 4 (Treewidth modulator)**.** A set $S \subseteq V(G)$ is a treewidth-η-modulator in a graph G if $\mathrm{tw}(G - S) \leq \eta$.

It is known that a treewidth-η-modulator S in a planar graph can be efficiently extended into a superset Z of bounded size whose removal separates the graph into components that interact with Z via constant-size neighborhoods. We cite one such statement from the book by Fomin et al. [15].

Lemma 2 ([15, Lemma 15.13])**.** *If a planar graph G has a treewidth-η-modulator S, then G has a set $Z \supseteq S$ such that*

- $|Z| \leq 4(\eta + 1)|S| + |S|$*, and*
- *each connected component of $G - Z$ has at most two neighbors in S and at most 2η neighbors in $Z \setminus S$.*

Furthermore, given G, S, and a tree decomposition (T, χ) of $G - S$, such a set Z can be computed in polynomial time.

We will derive the following useful consequence of this lemma. It tells us that we can use a treewidth modulator in a planar graph to efficiently find a decomposition of that graph with some desirable properties. The decomposition can be thought of as a weaker form of protrusion decomposition, using only elementary terminology.

Lemma 3. *For each fixed η there is a polynomial-time algorithm that, given a planar graph G and a treewidth-η-modulator S in G, computes a vertex set $Z \supseteq S$ such that*

- $|Z| = \mathcal{O}(\eta|S|)$*,*
- *the set $\{N_G(C) \mid C$ is a connected component of $G - Z\}$ has size $\mathcal{O}(\eta|S|)$, and*
- *for each connected component C of $G - Z$, we have $|N_G(C)| \leq 2\eta + 2$ and $|N_G(C) \cap S| \leq 2$.*

Proof: Given a planar graph G and treewidth-η-modulator S, the algorithm proceeds as follows. Since η is fixed, we can compute a tree decomposition (T, χ) of width η for G in time $\mathcal{O}(n)$ using Bodlaender's algorithm [5]. Using this tree decomposition, we invoke Lemma 2 to compute a superset $Z \supseteq S$ such that $|Z| \in \mathcal{O}(\eta|S|)$ and each connected component of $G - Z$ has at most two neighbors in S and at most 2η neighbors in $Z \setminus S$. Hence this choice of Z satisfies the first and third item in the lemma statement. It remains to argue that the second item is also satisfied, meaning that the number of distinct subsets of Z that occur as the open neighborhood of a connected component of $G - Z$, is bounded by $\mathcal{O}(\eta|S|)$. To derive this bound, we distinguish three cases based on the size of the neighborhood of a connected component C of $G - Z$.

1. For components C with $|N_G(C)| = 1$, there are trivially at most $|Z|$ distinct sets that can occur as $N_G(C)$ since $N_G(C) \subseteq Z$.
2. Now we bound the number of distinct open neighborhoods of connected components with a neighborhood of size two. Let $\mathcal{C}_2 = \{N_G(C) \mid C \text{ is a connected component of } G - Z \wedge |N_G(C)| = 2\}$. We bound $|\mathcal{C}_2|$ via an auxiliary graph H_2 on vertex set Z, which has an edge $\{u, v\}$ whenever there is a connected component C of $G - Z$ with $N_G(C) = \{u, v\}$. Observe that H_2 is a planar graph since it can be obtained as a minor from G by removing the components of $G - Z$ whose neighborhood size differs from two, while contracting the

components with a neighborhood of size two into a neighbor. Note that H_2 has a distinct edge for every element of $\mathcal{C}_2$. Hence $|\mathcal{C}_2| \leq |E(H_2)|$. Since H_2 is a planar graph on $|Z|$ vertices, it has at most $3|Z|$ edges by Euler's formula. Hence $|\mathcal{C}_2| \leq 3|Z|$.

3. Finally, we derive a bound for the neighborhoods of size at least three. Let $\mathcal{C}_{\geq 3} = \{N_G(C) \mid C \text{ is a connected component of } G - Z \wedge |N_G(C)| \geq 3\}$. Let $H_{\geq 3}$ be the auxiliary graph obtained from G as follows: contract each component of $G - Z$ whose neighborhood has size at least three, into a single vertex. Let $C_{\geq 3}$ denote the vertices that resulted from these contractions. Note that $|\mathcal{C}_{\geq 3}| \leq |C_{\geq 3}|$ since each open neighborhood is realized by at least one component. Each vertex of $C_{\geq 3}$ has at least three neighbors in $H_{\geq 3}$, and all its neighbors belong to Z since it was the result of contracting an entire connected component of $G - Z$. Hence we may apply Lemma 1 to $H_{\geq 3}$ to infer that $|\mathcal{C}_{\geq 3}| \leq |C_{\geq 3}| \leq 2|Z|$.

Since the three cases above are exhaustive, the number of distinct sets $N_G(C)$ that can arise from connected components C of $G - Z$ is bounded by $|Z| + 3|Z| + 2|Z| \in \mathcal{O}(|Z|) \leq \mathcal{O}(\eta|S|)$. This shows that the second item also holds and completes the proof of Lemma 3. □

3 Counting Minimum Feedback Vertex Sets

In this section, we explain the technique that allows us to either count the number of minimum feedback vertex sets of a graph G in polynomial time, or reduce G to a provably small instance with the same number of minimum solutions. We start by showing that, by using a few reduction rules, we can already reduce G to an equivalent instance with a specific structure. This reduction is based on the $\mathcal{O}(k^3)$-vertex kernel for the decision Feedback Vertex Set problem presented by Jansen [4]. We choose to use this kernel over the better-known and smaller-size kernels by Thomassé [32] and Iwata [19] because those rely on multiple reduction rules that are not safe for counting minimum solutions, while the one by Jansen has only one unsafe rule.

As is common for Feedback Vertex Set, we consider the graph we are working with to be undirected and we allow parallel edges. In this section, we make use of two reduction rules which are common for kernels of the decision variant of Feedback Vertex Set.

(R1) If there is an edge of multiplicity larger than two, reduce its multiplicity to two.
(R2) If there is a vertex v with degree at most one, remove v.

It can easily be verified that if an instance (G, k) is reduced to (G', k') by one of the rules above, we have $\#\text{minFVS}(G, k) = \#\text{minFVS}(G', k')$. Hence, these rules are safe in the context of counting minimum feedback vertex sets. Observe that if (R2) has exhaustively been applied on a graph G, then all vertices in $V_{\neq 2}(G)$ have degree at least three. For our purposes, we will need one additional method to reduce the graph. We first present a different lemma that will then motivate this third reduction rule.

Lemma 4. *Let X be a (not necessarily minimum) feedback vertex set of a graph G that is reduced with respect to (R2) and let $\mathcal{C}$ be the set of connected components of $G - V_{\neq 2}(G)$. Then it holds that*

(a) $|V_{\neq 2}(G)| \leq |X| + \sum_{v \in X} \deg_G(v)$, *and*
(b) $|\mathcal{C}| \leq |X| + 2\sum_{v \in X} \deg_G(v)$.

Proof: Consider the forest $F := G - X$. Partition the vertices of $V_{\neq 2}(G) \cap V(F) = V_{\neq 2}(G) \setminus X$ into sets $V'_{\leq 1}$, V'_2 and $V'_{\geq 3}$ for vertices that have respectively degree at most one, degree two or degree at least three in F. Furthermore, let V_ℓ denote the leaf nodes of F that have degree two in G. Since G has no vertices of degree at most one by (R2), the leaves of F are exactly the vertices $V_\ell \cup V'_{\leq 1}$. In any tree, the number of vertices of degree at least three is less than the number of leaves, thus $|V'_{\geq 3}| \leq |V'_{\leq 1}| + |V_\ell|$. Each vertex in V'_2 has at least one edge to X since they have degree at least three in G and degree exactly two in F. For a similar reason, each vertex in $V'_{\leq 1}$ has at least two edges to X. Each vertex in V_ℓ has one edge to X. Putting this together gives the following inequality, from which (a) directly follows.

$$\sum_{v \in X} \deg_G(v) \geq |V'_2| + 2|V'_{\leq 1}| + |V_\ell| \geq |V'_2| + |V'_{\leq 1}| + |V'_{\geq 3}| = |V_{\neq 2}(G) \setminus X| \tag{1}$$

To bound the size of the set $\mathcal{C}$ of connected components of $G - V_{\neq 2}(G)$, we instead bound the size of the set $\mathcal{C}'$ of connected components of $G - V_{\neq 2}(G) - X$. Observe that $|\mathcal{C}| \leq |\mathcal{C}'| + |X|$ since removing a vertex from a graph reduces the number of connected components by at most one. (Such a removal can *increase* the number of components by an arbitrary number, which is irrelevant for our argument.) Since X is an FVS, the connected components in $\mathcal{C}'$ can be seen as proper chains. Chains in $\mathcal{C}'$ that have both endpoints in F act as edges between those endpoints when it comes to the connectivity of F. This means that, since F is a forest, there can be at most $|V_{\neq 2}(G) \cap V(F)| \leq \sum_{v \in X} \deg_G(v)$ of such chains by Equation 1. All other chains will have at least one endpoint in X, which means there can be at most $\sum_{v \in X} \deg_G(v)$ of them, implying (b). $\square$

Based on Lemma 4, the goal of the third reduction rule is to decrease the degree of the vertices of a feedback vertex set. This idea is captured in Lemma 5. After presenting this lemma, we combine these results in Lemma 6 to create an algorithm to reduce a graph G to an, in context of counting minimum feedback vertex sets, equivalent graph with a bounded number of vertices of degree other than two and a bounded number of chains.

Lemma 5. *There exists a polynomial-time algorithm that, given a graph G reduced with respect to (R1), an integer k, a vertex $v \in V(G)$ and a feedback vertex set $Y_v \subseteq V(G) \setminus \{v\}$ of G, outputs a graph G' obtained by removing edges from G such that $\deg_{G'}(v) \leq |Y_v| \cdot (k+4)$ and $\#\text{minFVS}(G,k) = \#\text{minFVS}(G',k)$.*

Proof: Consider the forest $F := G - (Y_v \cup \{v\})$. For each $u \in Y_v$, mark trees of F that have an edge to both v and u until either all such trees are marked or at least $k+2$ of them are marked. Then, we construct a graph G' from G by removing all edges between v and trees of F that were not marked.

We shall first prove the bound on the degree of v in G'. The vertex v can have edges to vertices in Y_v or in F. Since (R1) has been exhaustively applied, there can be at most $2|Y_v|$ edges between v and Y_v. Each tree in F has at most one edge to v, since otherwise Y_v would not be an FVS of G. In G', only trees that were marked still have an edge to v, and since we mark at most $k+2$ trees per vertex in Y_v, we have at most $|Y_v| \cdot (k+2)$ of such trees. Combining this gives $\deg_{G'}(v) \leq |Y_v| \cdot (k+4)$.

To show that $\#\text{minFVS}(G,k) = \#\text{minFVS}(G',k)$, we prove that a vertex set $X \subseteq V(G)$ with $|X| \leq k$ is an FVS of G if and only if it is an FVS of G'. Clearly any FVS of G is an FVS of G' since G' is constructed from G by deleting edges. For the opposite direction, assume that X is an FVS of G' and assume for a contradiction that X is not an FVS of G. Then $G - X$ has a cycle W. This cycle must contain an edge $\{v,w\}$ of $E(G) \setminus E(G')$ since $G' - X$ is acyclic. By construction of G', we know that w is a vertex that belongs to an unmarked tree T. Since cycle W intersects T and since T is a connected component of $G - (Y_v \cup \{v\})$, there must be a vertex $u \in Y_v$ that has an edge to T. Since the edge between v and T is removed in G', there exist $k+2$ other trees that are marked and have an edge to both v and u. Since $|X| \leq k$, at least two of these trees are not hit by X. Furthermore, since v and u are part of W, they are also not contained in X. Therefore, there is a cycle in $G' - X$ through v, u and two of the aforementioned trees, contradicting that X is an FVS of G'. □

Lemma 6. *There is a polynomial-time algorithm that, given a graph G and integer k, outputs a graph G' and integer k' such that*

- $\#\text{minFVS}(G', k') = \#\text{minFVS}(G, k)$,
- $k' \leq k$,
- $|V_{\neq 2}(G')| = \mathcal{O}(k^3)$, *and*
- $G' - V_{\neq 2}(G')$ *has* $\mathcal{O}(k^3)$ *connected components.*

Proof: In our approach, we make use of the linear-time 4-approximation algorithm by Bar-Yehuda, Geiger, Naor, and Roth [3]. It can also approximate the more general problem of: for a given graph, find the smallest FVS that does not contain a given vertex. Our first step is to exhaustively apply (R1) on G and compute a 4-approximate FVS X of the graph. If $|X| > 4k$ then the feedback vertex number of G is larger than k, so we can return a trivial, constant size instance G' and k' such that $\#\text{minFVS}(G', k') = 0$. Otherwise, let G' and k' be a copy of G and k. For each vertex $v \in X$, compute a 4-approximate FVS Y_v of G' that does not contain v. If $|Y_v| > 4k$, then there does not exist a solution of size at most k that does not contain v, so remove v from G' and reduce k' by one. Otherwise, use Lemma 5 to reduce the degree of v in G'. Finally, we exhaustively apply reduction rule (R2) on G'.

Computing the 4-approximations and applying the reduction rules can be done in polynomial time. Each rule can only be applied a polynomial number of times, thus the algorithm runs in polynomial time. The fact that $\#\text{minFVS}(G, k) = \#\text{minFVS}(G', k')$ follows from safety of the reduction rules used in the algorithm and $k' \leq k$ follows from the fact that k' is never increased. We know that $|V_{\neq 2}(G')| = \mathcal{O}(k^3)$ and that $G' - V_{\neq 2}(G')$ has $\mathcal{O}(k^3)$ connected components due to Lemma 4 and the following bound:

$$\sum_{v \in X} \deg_{G'}(v) \leq \sum_{v \in X} |Y_v| \cdot (k+4) \leq \sum_{v \in X} 4k \cdot (k+4) = |X| \cdot 4k \cdot (k+4) = \mathcal{O}(k^3).$$

This concludes the proof of Lemma 6. □

The result of Lemma 6 is in and of itself not a proper kernel yet, since the chains of the graph it produces can be of arbitrary length. Our strategy to address this is as follows. If these chains are large in terms of k, then we can run an FPT algorithm in poly(n) time to count the number of minimum solutions. Otherwise, the chains can be replaced by structures of size poly(k) that do not change the number of minimum feedback vertex sets the instance has. This approach is captured in the following two lemmas and combined in the proof of Theorem 1.

Lemma 7. *There exists a polynomial-time algorithm that, given a graph G with a chain C and an integer k, outputs a graph G' obtained from G by replacing C with a vertex set C', and an integer k', such that*

- $\#\text{minFVS}(G', k') = \#\text{minFVS}(G, k)$,
- $G' - C' = G - C$,
- $N_{G'}(C') = N_G(C)$,

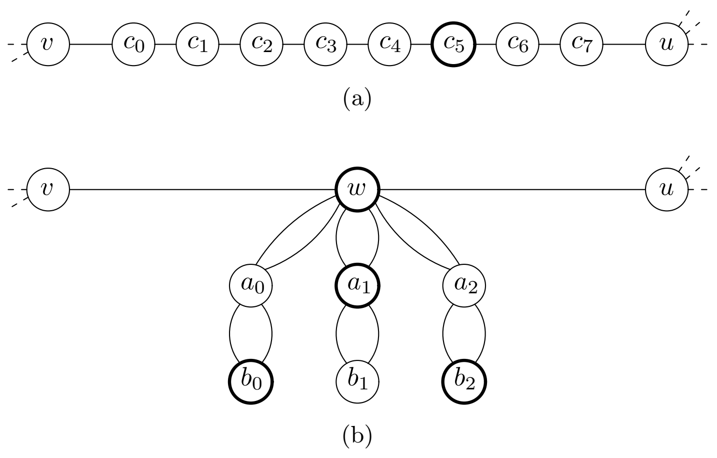


Figure 1: (a) A chain structure of size eight with two endpoints. (b) The replacement of the structure. An example of the mapping f is also illustrated through the vertices with a thicker border.

- $|C'| = \mathcal{O}(\log(|C|)^2)$, *and*
- $k' = k + \mathcal{O}(\log(|C|)^2)$.

Proof: In case C is a proper chain, the endpoints are defined as $N_G(C)$. If C is not a proper chain, which happens if C is a cycle in G, we choose an arbitrary vertex of C to act as its endpoint. For simplicity, we consider C to have two endpoints, where in some cases these two endpoints might be the same vertex.

First, we assume that the number of vertices of the chain, not including its endpoints, is a power of two, i.e. $|C| = 2^p$ for some integer p. Let $v, u \in V(G)$ be the endpoints of C (possibly $v = u$) and let $C = \{c_0, c_1, \cdots, c_{2^p-1}\}$. Then we construct our graph G' by replacing C by a gadget C', of which an example can be seen in Figure 1. It consists of the following elements.

- A vertex w with edges to both v and u.
- Pairs of vertices a_i, b_i for $0 \leq i < p$ such that there is an edge of multiplicity two between w and a_i and between a_i and b_i.

Additionally, we set $k' = k + p$.

We shall now prove that $\#\text{minFVS}(G, k) = \#\text{minFVS}(G', k')$. To this end, we define a mapping f from the set of minimum feedback vertex sets of G to those of G' and show that this is a bijection, which immediately implies that the two sets have the same cardinality. For a natural number m, define $\text{bin}(m)$ to be the binary representation of m on p bits and define $\text{bin}(m)_i$ to be the i'th least significant bit of $\text{bin}(m)$.

$$f(X) = \begin{cases} X \cup \{a_i \mid 0 \leq i < p\} & \text{if } X \cap C = \emptyset \\ \\ (X \setminus C) \cup \{w\} \cup \{a_i \mid 0 \leq i < p \wedge \text{bin}(m)_i = 0\} & \\ \cup \{b_i \mid 0 \leq i < p \wedge \text{bin}(m)_i = 1\} & \text{if } X \cap C = \{c_m\} \end{cases}$$

Note that for any minimum feedback vertex set X of G, we have $|X \cap C| \leq 1$ as picking any one vertex from C will already break all cycles that go through the chain.

▷ Claim 1. If a set X is a minimum FVS of G, then $f(X)$ is a minimum FVS of G'.

Proof We prove this in two parts. First we prove that $f(X)$ is an FVS of G' and then we prove that $f(X)$ is indeed an FVS of G' of minimum size.

To prove that $f(X)$ is an FVS of G', we use a proof by contradiction. Assume $f(X)$ is not an FVS of G', in which case a (simple) cycle W exists in $G' - f(X)$. This cycle must intersect C' as $X \setminus C = f(X) \setminus C'$ and $G - C = G' - C'$ thus $G - C - X = G' - C' - f(X)$, which means W would otherwise also exist in $G - X$, contradicting that X is an FVS of G. The cycle W cannot contain a b_j vertex since by definition of f, for $0 \leq i < p$, either a_i or b_i is in $f(X)$, which would either mean b_j is isolated in $G' - f(X)$ or is removed. Also, W can not contain any a_j vertex as, by definition of f, if a_j is not in $f(X)$, then both its neighbors w and b_j are in $f(X)$. This leaves only the option that W intersects C' through only vertex w. This means that $W - \{w\}$ contains a path from v to u in $G' - f(X) - \{w\}$. However, since as mentioned before $G - C - X = G' - C' - f(X)$, this same path also exists in $G - X$. Furthermore, since $w \notin f(X)$, that must mean that $X \cap C = \emptyset$ so $(W \setminus \{w\}) \cup C$ would form a cycle in $G - X$ contradicting that X is an FVS of G.

Next we prove that $f(X)$ is a minimum FVS of G'. Assume for sake of a contradiction that $f(X)$ is not a minimum FVS of G' due to the existence of a $X' \subseteq V(G')$ with $|X'| < |f(X)|$ such that X' is an FVS of G'. We first observe that any FVS of G' contains at least p vertices from C' since all p of the pairs (a_i, b_i) form vertex-disjoint cycles. Similarly so, if an FVS of G' contains w, then it contains at least $p+1$ vertices from C'. We distinguish two cases.

Case $w \notin X'$**:** Then $X' \setminus C'$ is an FVS of G. If $G - (X' \setminus C')$ would contain a cycle, then there would also exist a cycle in $G' - X'$ since $G - C = G' - C'$ and both graphs $G - (X' \setminus C')$ and $G' - X'$ have a vu path, the former through C and the latter through w. Furthermore, $|X' \setminus C'| \leq |X'| - p < |f(X)| - p = |X| + p - p = |X|$, contradicting that X is a minimum FVS of G.

Case $w \in X'$**:** Then $Y := (X' \setminus C') \cup \{c_j\}$ is an FVS of G for any $0 \leq j < 2^p$. If this were not the case, then, since Y contains a chain vertex, a cycle would need to exist completely in $G - C - Y$ which is the same graph as $G' - C' - X'$. This would contradict X' being an FVS of G'. Furthermore, $|Y| \leq |X'| - (p+1) + 1 < |f(X)| - p = |X| + p - p = |X|$, contradicting that X is a minimum FVS of G.

As both cases lead to a contradiction, we conclude that $f(X)$ is a minimum FVS of G'. ◁

▷ Claim 2. The function f is bijective.

Proof We first argue that f is injective. Let X and Y be two minimum feedback vertex sets of G such that $X \neq Y$. That means $X \setminus C \neq Y \setminus C$ or $X \cap C \neq Y \cap C$. The former immediately allows us to conclude that $f(X) \neq f(Y)$ since $X \setminus C = f(X) \setminus C'$ and $Y \setminus C = f(Y) \setminus C'$. In the second case, we have two options. The first is that $X \cap C = \{c_j\}$ and $Y \cap C = \{c_m\}$ for some $j \neq m$, and since binary representations are unique, they lead to different sets $f(X)$ and $f(Y)$. The second option is that either X or Y contains no vertex from C while the other one does. Then only one of $f(X)$ or $f(Y)$ contains w and the other one does not, so $f(X) \neq f(Y)$.

It remains to show that f is surjective. To this end, we take an arbitrary minimum FVS X' of G' and show that there exists a minimum FVS X of G such that $X' = f(X)$. We distinguish two cases.

Case $w \notin X'$**:** For this case, first observe that $\{a_i \mid 0 \leq i < p\} \subseteq X'$ since otherwise w would form a cycle with one of these a_i vertices in $G - X'$. Furthermore, $b_i \notin X'$ for $0 \leq i < p$

since X' already contains a_i, leaving b_i isolated in $G - X'$ and thus a redundant choice for a minimum FVS. From this we can conclude that $X' \cap C' = \{a_i \mid 0 \leq i < p\}$. Therefore, taking $X := X' \setminus C'$ would give $f(X) = X'$. We have already argued in case $w \notin X'$ that $X' \setminus C'$ is an FVS of G. To show that it is a minimum FVS of G, note that if there was a $Y \subseteq V(G)$, $|Y| < |X|$ such that Y is an FVS of G, then $f(Y)$ is an FVS of G' and $|f(Y)| = |Y| + p < |X| + p = |X' \setminus C'| + p = |X'| - p + p = |X'|$ which would contradict X' being a minimum FVS of G'.

Case $w \in X'$: For this case, we instead observe that for each pair $\{a_i, b_i\}$, exactly one of $\{a_i, b_i\}$ is in X', as otherwise X' would not be an FVS of minimum size. Since there are p of such pairs, there is a unique value $0 \leq j < 2^p$ such that the binary representation of j corresponds to the choice of a and b vertices in X'. We can then choose $X := (X' \setminus C') \cup \{c_j\}$. We clearly have that $f(X) = X'$. Also, we have already shown before that X is an FVS of G and the argument that X is a minimum FVS of G is analogous to that in case $w \notin X'$.

From this reasoning we can conclude that f is a bijective function from the set of minimum feedback vertex sets of G to the set of minimum feedback vertex sets of G'. $\lhd$

In the argument above, we assumed that the length of the chain is a power of two. We can address this by noting that any natural number can be written as a sum of unique powers of two. Similarly, we can decompose the chain C into a number of subpaths each having a number of vertices that is a unique power of two. We can then apply the replacement described above on each subpath individually to get multiple replacement structures in a chain between the endpoints of C. As seen before, the size of the replacement is linear in the exponent of the length of the chain. In the worst case, when expressing $|C|$ in binary as a sum of distinct powers of two, the exponents of these powers sum up to $\mathcal{O}(\log(|C|)^2)$, which is also the bound on the number of vertices used in our replacement and on the increase in the parameter value. $\square$

We remark for Lemma 7 that a similar chain replacement gadget without parallel edges can be constructed, at the expense of a linear increase in the size of the gadget. Note that the gadget described in the lemma fails to preserve the number of *inclusion-minimal* feedback vertex sets. While each inclusion-minimal FVS of size k in graph G maps to an inclusion-minimal FVS of size $k+p$ in graph G', the converse fails. If we take an FVS X' of G' that contains vertex w together with $\{b_0, \ldots, b_{p-1}\}$, then the corresponding FVS X in G obtained by replacing $\{w, b_0, \ldots, b_{p-1}\}$ by a single vertex c_i on the chain might fail to be inclusion-minimal. This happens if the only cycles through w in the graph $G' - (X' \setminus \{w\})$ are those involving the a_i-vertices. Since they are not present in G, it might be that there are no cycles through c_i in the graph $G - (X \setminus \{c_i\})$, so that $X \setminus \{c_i\}$ is a strict subset that is still an FVS, violating inclusion-minimality. Hence the gadget cannot be used for the task of counting inclusion-minimal feedback vertex sets.

By adapting the iterative compression algorithm by Cao, Chen, and Liu [9] for the decision Feedback Vertex Set problem, we can derive the following lemma. Its proof is given in Appendix A.

Lemma 8. *There is an algorithm that, given a graph G and integer k, computes $\#\mathrm{minFVS}(G, k)$ in $2^{\mathcal{O}(k)} \cdot \mathrm{poly}(n)$ time.*

Using the machinery constructed so far, we can prove our main result for Feedback Vertex Set.

Theorem 1. *There is a polynomial-time algorithm that, given a graph G and integer k, either*

- *outputs* $\#\text{minFVS}(G,k)$*, or*
- *outputs a graph* G' *and integer* k' *such that* $\#\text{minFVS}(G,k) = \#\text{minFVS}(G',k')$ *and* $|V(G')| = \mathcal{O}(k^5)$ *and* $k' = \mathcal{O}(k^5)$*.*

Proof: First we use the algorithm from Lemma 6 to find a graph G^* and integer $k^* \leq k$ such that $\#\text{minFVS}(G,k) = \#\text{minFVS}(G^*,k^*)$, $|V_{\neq 2}(G^*)| = \mathcal{O}(k^3)$ and $G^* - V_{\neq 2}(G^*)$ has $\mathcal{O}(k^3)$ connected components (chains). Then, if there is a chain of size larger than 2^k, we can run the algorithm from Lemma 8 to compute $\#\text{minFVS}(G^*,k^*)$ in poly(n) time since $n > 2^k$. Otherwise, all chains of G^* have size at most 2^k and we can use Lemma 7 on G^* to find a graph G' and integer k' such that

- $\#\text{minFVS}(G',k') = \#\text{minFVS}(G^*,k^*) = \#\text{minFVS}(G,k)$,
- $|V(G')| = \mathcal{O}(k^3) + \mathcal{O}(k^3 \cdot \log(2^k)^2) = \mathcal{O}(k^5)$, and
- $k' = \mathcal{O}(k + k^3 \cdot \log(2^k)^2) = \mathcal{O}(k^5)$.

This concludes the proof of Theorem 1. □

4 Counting Minimum Dominating Sets in Planar Graphs

In this section, we show how our approach can also be used through a different technique. In Section 3 we adapted a decision kernel and replaced remaining, potentially large structures by smaller, similarly behaving structures. The general outline of our tactic in this section is to decompose the graph in a 'small' core and a 'small' number of subgraphs protruding from this core. We then continue to shrink and replace these subgraphs, all while making sure we do not change the answer to the counting problem.

We consider the problem of counting minimum dominating sets of a planar graph. In the context of dominating sets, we assume all graphs to be undirected and simple, so they do not have parallel edges. We start by showing that if a planar graph G has a dominating set of size at most k, we can use this to decompose G.

Lemma 9. *There is a polynomial-time algorithm that, given a planar graph* G *and an integer* k*, either concludes that* G *does not have a dominating set of size at most* k*, or outputs a dominating set* S *of size at most* $2k$ *and a vertex set* $Z \supseteq S$ *of size* $\mathcal{O}(k)$ *such that*

- *the set* $\{N_G(C) \mid C \text{ is a connected component of } G - Z\}$ *has size* $\mathcal{O}(k)$*, and*
- *for each connected component* C *of* $G - Z$*, we have that* $|N_G(C)| \leq 6$ *and* $|N_G(C) \cap S| \leq 2$*.*

Proof: The decision version of Dominating Set on planar graphs admits a polynomial-time approximation scheme [2]. As a first step, we can use this to find a 2-approximate dominating set S of G in polynomial time. If $|S| > 2k$ then the domination number of G is greater than k, so we are done. Otherwise, in case G is connected, we can turn S into a connected dominating set S' of G such that $|S'| \leq 3|S| \leq 6k$ [15, Claim 15.10]. If G is not connected, we can instead do the same for each connected component of the graph individually. Since G is a planar graph, $G - S'$ is an outerplanar graph (cf. [15, Lemma 15.9]) and thus has treewidth at most two [6, Lemma 78]. This means that S' is a treewidth-2-modulator for G. Therefore, we can use Lemma 3 to find in polynomial time a set Z that has the properties we require. □

For our purpose, it might be useful to also consider the properties of Lemma 9 in different terms. The first property tells us that when we group the connected components of $G - Z$ based on their open neighborhood in G, we end up with only $\mathcal{O}(k)$ groups. By the second property, if we let P be the vertices of any one such group of connected components, then the vertex set $N_G(P) \cap S$ consists of at most two vertices, yet completely dominates P. We shall refer to such a set P as a *protrusion* of G with respect to Z (see Definition 5 for a formal definition). In the remainder of this section we will focus on how we can replace these protrusions without changing the number of minimum dominating sets.

Definition 5 (Protrusion)**.** Let G be a graph and let $Z \subseteq V(G)$. We say that a non-empty vertex set P is a protrusion of G with respect to Z if $P = \{v \in C \mid C$ is a connected component of $G - Z$ and $N_G(C) = U\}$ for some $U \subseteq Z$.

Hence a protrusion with respect to Z is an inclusion-maximal set of vertices contained in connected components of $G - Z$ that all have the same open neighborhood U. Given Z, one can compute a partition of $V(G - Z)$ into protrusions in polynomial time by grouping the connected components of $G - Z$ based on their open neighborhood. We remark that our notion of protrusion here is inspired by, but distinct from, the notion of r-protrusions used in the meta-kernelization framework [7]. Our ad-hoc formulation for the Dominating Set problem avoids the use of tree decompositions, which makes it easier to work with. We will eventually use the decomposition into protrusions to obtain our counting kernelization. But before doing so, we first analyze the properties of one particular type of protrusion, introduced below.

We call a vertex set $C \subseteq V(G)$ a *wide diamond* if there are distinct vertices $v, u \in V(G) \setminus C$ such that for all $c \in C$, we have that $N_G(c) = \{v, u\}$. We consider v and u to be the endpoints of the wide diamond C. We make the following observation concerning minimum dominating sets on wide diamonds.

Observation 1. *If C is a wide diamond with distinct endpoints v, u in a graph G and $|C| \geq 3$, then any minimum dominating set X of G satisfies the following properties:*

1. *$X \cap \{v, u\} \neq \emptyset$, since if X contains neither v nor u, we must have that $C \subseteq X$, while $(X \setminus C) \cup \{v, u\}$ would then be a dominating set of smaller size.*
2. *If $\{v, u\} \subseteq X$, then $X \cap C = \emptyset$, since $N_G[c] \subseteq N_G[\{v, u\}]$ for all $c \in C$.*
3. *$|X \cap C| \leq 1$, since by Property 1, we can assume w.l.o.g. that $v \in X$, and $N_G[c] \setminus N_G[v] = \{u\}$ for all $c \in C$.*

In Lemma 10 we show how we can efficiently replace a wide diamond by a smaller, similarly behaving structure. We show how we can use this replacement to reduce the protrusions of our decomposition of the graph in Lemma 11.

Lemma 10. *There exists a polynomial-time algorithm that, given a planar graph G with a wide diamond C and an integer k, outputs a planar graph G' obtained from G by replacing C with a vertex set C', and an integer k', such that*

- $\#\text{minDS}(G', k') = \#\text{minDS}(G, k)$,
- $G' - C' = G - C$,
- $N_{G'}(C') = N_G(C)$,
- $|C'| = \mathcal{O}(\log(|C|)^2)$, *and*
- $k' = k + \mathcal{O}(\log(|C|)^2)$.

Proof: Let vertices v and u be the endpoints of the wide diamond C. Assume without loss of generality that C consists of at least five vertices, since the statement is trivial otherwise. For

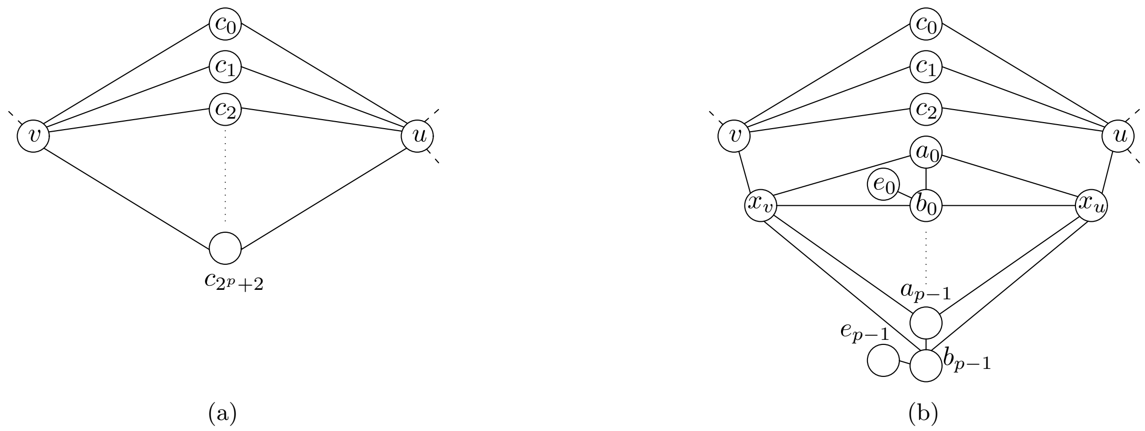


Figure 2: (a) A wide diamond structure with endpoints. (b) The replacement of the structure.

simplicity, we first assume that $|C| = 2^p + 3$ for some integer p, so let $C = \{c_0, c_1, \cdots, c_{2^p+2}\}$. We then construct G' from G by replacing C by C', which has the following structure. In C', we leave three vertices c_0, c_1, c_2 from C untouched and replace all others.

- Add vertex x_v with an edge to v and vertex x_u with an edge to u.
- For each $0 \le i < p$, add two adjacent vertices a_i and b_i and make these vertices adjacent to both x_v and x_u. Add a degree-one vertex e_i adjacent to b_i.

It is easy to see that G' is planar since the constructed gadget has a planar embedding with its only boundary vertices x_v, x_u on a single face (see Figure 2). The gadget can be drawn into any face of an embedding of G that contains both u and v, whose existence is guaranteed because $N_G(c_0) = \{v, u\}$. For the parameter, we set $k' = k + p$.

We define a function f from the set of minimum dominating sets of G to the set of minimum dominating sets of G' and we prove that $\#\text{minDS}(G, k) = \#\text{minDS}(G', k')$ by showing that f is a well-defined, bijective function.

$$f(X) = \begin{cases} X \cup \{b_i \mid 0 \le i < p\} & \text{if } X \cap C \subseteq \{c_0, c_1, c_2\} \\ \\ (X \setminus C) \cup \{x_u\} \cup \{b_i \mid 0 \le i < p \wedge \text{bin}(m-3)_i = 0\} & \\ \cup \{e_i \mid 0 \le i < p \wedge \text{bin}(m-3)_i = 1\} & \text{if } X \cap C = \{c_m\} \wedge m \ge 3 \wedge v \in X \\ \\ (X \setminus C) \cup \{x_v\} \cup \{b_i \mid 0 \le i < p \wedge \text{bin}(m-3)_i = 0\} & \\ \cup \{e_i \mid 0 \le i < p \wedge \text{bin}(m-3)_i = 1\} & \text{if } X \cap C = \{c_m\} \wedge m \ge 3 \wedge u \in X \end{cases}$$

Note that since we assumed $|C| \ge 5$, by Observation 1 the function f covers all cases for a minimum DS X of G. Also note that $\{c_0, c_1, c_2\}$ still forms a wide diamond of size at least three between v and u, so the properties of the observation still apply in G'.

▷ Claim 3. If a set X is a minimum DS of G, then $f(X)$ is a minimum DS of G'.

Proof We prove this claim in two parts. We first prove that $f(X)$ is a DS of G', and then we prove that it is a DS of minimum size.

To prove that $f(X)$ is a DS of G', we use a proof by contradiction, so assume $f(X)$ is not a DS of G'. This means there is a vertex $w \in V(G')$ that is not dominated by $f(X)$. If w is a vertex of $G' - C' - \{v, u\}$, then since $G' - C' = G - C$, we have that w is also a vertex of $G - C - \{v, u\}$. Furthermore, since $X \setminus C = f(X) \setminus C'$ and $N_G(C) = \{v, u\}$, if w is not dominated by $f(X)$ it is also not dominated by X, which contradicts X being a dominating set of G. Thus $w \in C'$ or $w \in \{v, u\}$. The former can never be true, as in all cases the function f takes the union over a dominating set of $C' \setminus \{c_0, c_1, c_2\}$ and by Property 1 of Observation 1, either v or u is in X, and thus also in $f(X)$, ensuring that c_0, c_1 and c_2 are dominated. So, assume without loss of generality that $w = u$. Then, by Property 1 of Observation 1, that means $v \in X$ and since u is dominated by X in G that $X \cap C = \{c_m\}$ for some c_m. However, by definition of f, that would mean that either $c_m \in f(X)$ or $x_u \in f(X)$ and both c_m and x_u have an edge to u which contradicts that u is not dominated by $f(X)$. Thus $f(X)$ is a dominating set of G'.

We also use a proof by contradiction to show that $f(X)$ is a minimum DS of G'. Assume that $f(X)$ is not a minimum DS of G' due to the existence of a set $X' \subseteq V(G')$ with $|X'| < |f(X)|$ such that X' is a dominating set of G'. Assume that X' is a minimum such dominating set. We distinguish three cases.

Case $x_v, x_u \notin X'$: We can then show that $\{b_i \mid 0 \leq i < p\} \subseteq X'$. If there would be a $b_j \notin X'$, then since also $x_v, x_u \notin X'$, we would need to have that $a_j, e_j \in X'$ since these are the only remaining options to dominate these vertices. However, since $N_{G'}[\{a_j, e_j\}] = N_{G'}[b_j]$, taking $(X' \setminus \{a_j, e_j\}) \cup \{b_j\}$ would result in a dominating set of smaller cardinality, contradicting that X' is a minimum dominating set of G'. Thus $\{b_i \mid 0 \leq i < p\} \subseteq X'$. From this it follows that $Y := X' \setminus \{b_i \mid 0 \leq i < p\}$ is a dominating set of G. We already have $v \in X'$ or $u \in X'$ by Property 1 of Observation 1, ensuring that all vertices in C are dominated. Also, since $x_v, x_u \notin X'$, the set X' dominates u and v from $V(G) \setminus (C \setminus \{c_0, c_1, c_2\})$. Furthermore, $|Y| = |X'| - p < |f(X)| - p = |X| + p - p = |X|$, contradicting that X is a minimum DS of G.

Case $x_v \in X'$: As a first step, we can realize that $v \notin X'$. If v would be in X', then we could construct a DS of G' of smaller size than X' by taking $(X' \setminus C') \cup \{b_i \mid 0 \leq i < p\}$ (potentially also including x_u if $x_u \in X'$). Thus, $v \notin X'$ which by Property 1 of Observation 1 implies that $u \in X'$. Knowing this, we consider $Y := (X' \setminus C') \cup \{c_j\}$ for some $0 \leq j < 2^p + 3$. This set Y is a dominating set of G since all vertices $G - C - \{v, u\}$ are unaffected by the change, $u \in Y$ which ensures all vertices c_i are dominated, and $c_j \in Y$ ensuring that v is dominated. Notice that $|X' \cap C'| \geq p+1$ since $x_v \in X'$ and X' must contain either e_i or b_i for each $0 \leq i < 2^p$ in order to dominate e_i. Thus, $|Y| \leq |X'| - (p+1) + 1 = |X'| - p < |f(X)| - p = |X| + p - p = |X|$, contradicting that X is a minimum DS of G.

Case $x_u \in X'$: The argument for this case is symmetric to that of the previous case.

In all cases we reach a contradiction from which we can conclude that $f(X)$ is a minimum DS of G'. ◁

From this we can conclude that f is a well-defined mapping from the set of minimum dominating sets of G to those of G'.

▷ Claim 4. The function f is bijective.

Proof The function f is clearly injective as when $X \neq Y$, either $X \setminus C \neq Y \setminus C$ or $X \cap C \neq Y \cap C$ which both imply that $f(X) \neq f(Y)$ by definition of f. Following the same type of argumentation as in Claim 3, it is easy to verify the following. For any minimum dominating set X' in G', if we define X as follows:

$$X = \begin{cases} X' \setminus \{b_i \mid 0 \leq i < p\} & \text{if } X' \cap \{x_v, x_u\} = \emptyset \\ (X' \setminus C') \cup \{c_j\} & \text{if } X' \cap \{x_v, x_u\} \neq \emptyset \end{cases}$$

then X is a minimum DS of G and $f(X) = X'$ for the correct choice of j, which is determined by which b and e vertices are in X'. Thus the function is surjective, and therefore also a bijection. $\lhd$

Similar as for counting feedback vertex sets, we can address the issue that we assumed $|C| = 2^p + 3$ for some p by decomposing the wide diamond C into multiple smaller wide diamonds such that one consists of three vertices and all others consist of a number of vertices that is a unique power of two. In order for the complete structure to get the behavior we desire, it suffices to not replace the one wide diamond consisting of three vertices to let this act as c_0, c_1, c_2 in the replacement procedure for all other replacements. We then replace all other wide diamonds entirely in the way described above. This allows us to replace any wide diamond C by a structure with $\mathcal{O}(\log(|C|)^2)$ vertices, and $k' = k + \mathcal{O}(\log(|C|)^2)$. $\square$

The next lemma shows that the previous replacement procedure for wide diamonds can be used more generally to shrink large protrusions (Definition 5) without changing the number of minimum dominating sets of size k.

Lemma 11. *There is a polynomial-time algorithm that, given a planar graph G, integer k, vertex set Z and protrusion P of G with respect to Z such that $|N_G(P)| \leq 6$ and P can be dominated by a set of at most 2 vertices from $N_G(P)$, does the following. The algorithm outputs a planar graph G' obtained from G by replacing P by a vertex set P', and integer k', such that*

- $\#\text{minDS}(G', k') = \#\text{minDS}(G, k)$,
- $G' - P' = G - P$,
- $N_{G'}(P') = N_G(P)$,
- $|P'| = \mathcal{O}(\log(|P|)^2)$, *and*
- $k' = k + \mathcal{O}(\log(|P|)^2)$.

Proof: The proof is structured as follows. We will first remove certain edges and vertices in the protrusion that are not relevant in the context of dominating sets. We then prove that the part of the protrusion that remains can only have $\mathcal{O}(1)$ vertices that are not part of a wide diamond. Finally, we use Lemma 10 to replace these wide diamonds.

Consider a minimum dominating set X of G. The first thing we note is that X contains at most 6 vertices from P. This is because $N_G(P)$ dominates P with $|N_G(P)| \leq 6$ and for all $D \subseteq P$, we have that $N_G[D] \subseteq N_G[N_G(P)]$. Thus, if X would contain more than 6 vertices of P, then $(X \setminus P) \cup N_G(P)$ would be a dominating set of G of smaller size, contradicting that X is a minimum dominating set. Additionally, we note that clearly the vertex set $X \cap N_G[P]$ dominates P. Based on these observations, the goal of our reduction is to preserve these sets of at most 6 vertices of $N_G[P]$ that dominate P.

The first step will be to remove redundant edges between vertices of P. More formally, an edge $\{v, u\}$ with $v, u \in P$ is called *redundant* if for each set D of size at most 6 that dominates P, the set D still dominates P when we remove edge $\{v, u\}$. We can naively check if an edge is redundant by enumerating all subsets of $N_G[P]$ of size at most 6 that dominate P in poly(n) time and checking if removing the edge affects them. Next, we remove certain vertices from P using the following observation. If there is a vertex $v \in V(G)$ with more than two neighbors of degree one, then all minimum dominating sets of G will contain v and none of those degree-one neighbors. Therefore, removing all but two of the degree-one neighbors does not affect the set of minimum dominating sets of G. After we have removed redundant edges from P, we use this observation to remove unnecessary degree-one vertices from P.

Assume now that we used the techniques described above to remove edges and vertices from P. We can analyze the structure that the remaining graph $G[N_G[P]]$ has by counting the types of vertices that appear in it. We first consider two edge cases.

If $N_G(P) = \emptyset$ then also $P = \emptyset$ because $N_G(P)$ is a dominating set of P. If $N_G(P) = \{s\}$ for some vertex s, then we can always find a replacement P' for P that has only a constant number of vertices. For this, we first realize that, since $\{s\}$ dominates P in this specific case, any minimum dominating set X of G contains at most one vertex from $N_G[P]$. Therefore, X can only ever contain a vertex from P that dominates all vertices in P. Let U be the set of vertices of P that individually dominate P. If $U = \emptyset$, then, since $P \neq \emptyset$, any minimum dominating set of G must contain s. Thus we can let P' consist of only s and two new vertices, both having only an edge to s. If $U \neq \emptyset$, then we construct P' from P by removing all vertices other than s and those in U. There can be at most three vertices in U, since otherwise $G[U \cup \{s\}]$ would form a clique of size at least 5, which would contradict G being planar by Theorem 2. To see why this reduction is safe, note that for any minimum dominating set X of G, we must have that $X \cap N_G[P] = \{v\}$ for some vertex $v \in U \cup \{s\}$. If $|X \cap N_G[P]| > 1$, then X is not of minimum size, and if $v \notin U \cup \{s\}$, then there is a vertex in P not dominated by X.

From this point on, assume $|N_G(P)| > 1$ and let s_1, s_2 be two distinct vertices in $N_G(P)$ that together dominate P. We partition the vertices of P into sets $P_{\leq 1}$, P_2 and $P_{\geq 3}$ based on whether their degree is at most one, is two, or is at least three, respectively. We first look at $P_{\leq 1}$.

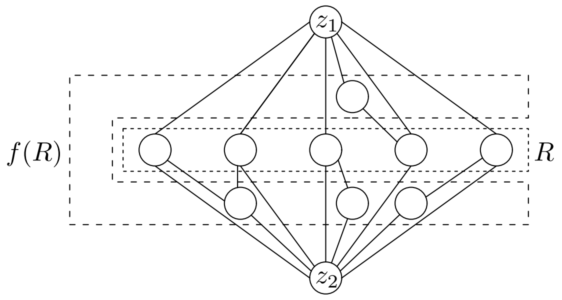
Note that since $\{s_1, s_2\}$ dominates P, all vertices of $P_{\leq 1}$ must be incident to exactly one edge and this edge has as other endpoint either s_1 or s_2. There cannot be more than two degree-one neighbors per vertex, since then we would have removed them. From this we can conclude that $|P_{\leq 1}| \leq 4$.

Before we look at $P_{\geq 3}$, we will first prove the following claim to be correct. This claim will in turn be useful for bounding the number of vertices contained in $P_{\geq 3}$.

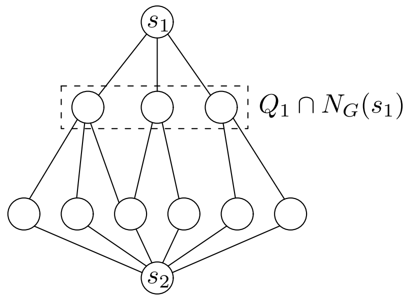
$\triangleright$ Claim 5. If P has no redundant edges, then for each distinct pair of vertices $z_1, z_2 \in N_G[P]$, there can be at most 72 vertices in $N_G(z_1) \cap N_G(z_2) \cap P$ that are adjacent to at least one (not necessarily the same) vertex in $P \setminus \{z_1, z_2\}$.

Proof Let $R \subseteq N_G(z_1) \cap N_G(z_2) \cap P$ be the set of vertices that are adjacent to both z_1 and z_2 and at least one vertex in $P \setminus \{z_1, z_2\}$. Assume for a contradiction that $|R| > 72$. We first prove that all vertex sets of size at most 6 that dominate P must contain at least one of $\{z_1, z_2\}$. If this were not the case, then there exists a dominating set D with $|D| \leq 6$ such that $z_1, z_2 \notin D$. All vertices in R must be dominated by D, so there is a vertex $v \in D$ that dominates more than $72/6 = 12$ vertices of R. This means that v has an edge to more than 11 vertices in R. However, then v, z_1, z_2 and three of those vertices of R adjacent to v would together form $K_{3,3}$, which contradicts G being planar by Theorem 2. Thus, each set of size 6 that dominates P must contain z_1 or z_2.

We associate a vertex from $P \setminus \{z_1, z_2\}$ to each vertex of R through a function $f \colon R \to P \setminus \{z_1, z_2\}$, where for each vertex $v \in R$, we let $f(v)$ be an arbitrary vertex in $N_{G[P]}(v) \setminus \{z_1, z_2\}$. Next, let $f(R) := \{f(v) \mid v \in R\}$ be the set of vertices of $P \setminus \{z_1, z_2\}$ that are associated to some vertex of R. See Figure 3a for an example of the vertices involved in the structure under consideration. No vertex in $f(R)$ has an edge to both z_1 and z_2, since then all edges from R to that vertex would have been redundant due to z_1 or z_2 being in every set of size at most 6 that



(a)



(b)

Figure 3: (a) An example of the types of vertices involved in the proof of Claim 5. (b) An example of the types of vertices involved in the proof of Claim 7.

dominates P.

Furthermore, each vertex in $f(R)$ can be the image of at most two vertices of R. If there were distinct $a, b, c \in R$ such that $f(a) = f(b) = f(c)$, then $\{z_1, z_2, f(a)\}$ and $\{a, b, c\}$ would together form $K_{3,3}$. Thus, since $|R| > 72$, we must have $|f(R)| > 36$. Each vertex in $f(R)$ is non-adjacent to at least one of $\{z_1, z_2\}$, so at least half of them are non-adjacent to the same one. Assume without loss of generality that at least half of them, so more than 18, are non-adjacent to z_1. Let $U \subseteq f(R)$ refer to this set of vertices. We consider two cases.

Case: There exists a set $D \subseteq N_G[P] \setminus \{z_2\}$ of size at most 6 that dominates P. Naturally, D also dominates $U \subseteq P$. Since $z_2 \notin D$ and z_1 has no edges to U, the set $D \setminus \{z_1, z_2\}$ must also dominate U. Since this set contains at most 6 vertices, there must be a vertex $v \in D \setminus \{z_1, z_2\}$ that dominates more than $18/6 = 3$ vertices of U and thus has an edge to three distinct vertices $f(a), f(b), f(c) \in U$ for some $a, b, c \in R \setminus \{v\}$. However, this would contradict G being planar due to it containing $K_{3,3}$ as a minor. One side of the minor consists of $\{z_1, z_2, v\}$, and the other side consists of the vertices obtained by contracting the edges $\{a, f(a)\}$, $\{b, f(b)\}$ and $\{c, f(c)\}$.

Case: All sets of size at most 6 that dominate P contain z_2. If z_2 has no edge to any vertex in U, then the reasoning of the previous case still applies. If z_2 does have an edge to some $f(u) \in U$ for $u \in R$, then this would imply that the edge $\{u, f(u)\}$ is redundant, another contradiction.

Since both cases lead to a contradiction, this completes the proof of the claim. ◁

Having proven this claim, we shall now consider $P_{\geq 3}$, which we partition further into sets Q_1, Q_2 and $Q_{\geq 3}$ based on whether they have one, two, or at least three edges to $N_G(P)$, respectively. We will prove that $P_{\geq 3}$ has constant size by proving that all three of these sets have constant size. We can immediately conclude that $|Q_{\geq 3}| = \mathcal{O}(1)$ using Lemma 1 since G is planar, as by definition each vertex in $Q_{\geq 3}$ has at least three neighbors in $N_G(P)$ and $|N_G(P)| \leq 6$.

▷ Claim 6. The set Q_2 consists of at most $15 \cdot 72$ vertices.

Proof The set Q_2 consists of the vertices of P that have degree at least three and have exactly two edges to $N_G(P)$. Thus, all vertices of Q_2 exist between pairs of $N_G(P)$ vertices. Consider an arbitrary pair of distinct vertices $z_1, z_2 \in N_G(P)$. Then, since by definition each vertex in Q_2

has an edge to at least one vertex not in $N_G(P)$, we can use Claim 5 to conclude that $|N_G(z_1) \cap N_G(z_2) \cap Q_2| \leq 72$. Since there are at most $\binom{6}{2} = 15$ pairs of $N_G(P)$ vertices, we arrive at the bound stated in the claim. $\lhd$

$\rhd$ Claim 7. The set Q_1 consists of at most $2 \cdot 6 \cdot 73$ vertices.

Proof The set Q_1 consists of the vertices of P that have degree at least three and have exactly one edge to $N_G(P)$. Note that since $\{s_1, s_2\}$ dominates the vertices in P, all vertices in Q_1 have an edge to either s_1 or s_2 and no other vertices in $N_G(P)$. To bound the number of vertices in Q_1, we will bound the number of such vertices that can be adjacent to s_1, and realize that the same bound applies for s_2. First observe that if all sets of size at most 6 that dominate P contain both s_1 and s_2, then all edges between vertices of P would have been redundant and we could immediately conclude that $|Q_1| = 0$. Thus we consider this not to be the case. We shall prove that $|Q_1 \cap N_G(s_1)| \leq 6 \cdot 73$ using a proof by contradiction. To this end assume $|Q_1 \cap N_G(s_1)| > 6 \cdot 73$. We distinguish two cases.

Case: There exists a set of size at most 6 that dominates P and does not contain s_1. That means that all vertices in $N_G(s_1) \cap P$ can be dominated with at most 6 vertices other than s_1. Since $|Q_1 \cap N_G(s_1)| > 6 \cdot 73$ there must be a vertex v that dominates more than 73 vertices in $Q_1 \cap N_G(s_1)$ and thus has an edge to more than 72 vertices of them (note v could be in $Q_1 \cap N_G(s_1)$). We have $v \in P$, since the dominating set avoids s_1 which is the only neighbor of $Q_1 \cap N_G(s_1)$ among $N_G(P)$. Since the vertices in Q_1 have degree at least three, they must all have an edge to at least one vertex in $P \setminus \{s_1, v\}$. This means we can use Claim 5 to conclude that $|N_G(s_1) \cap N_G(v) \cap Q_1| \leq 72$, which is a contradiction.

Case: All sets of size at most 6 that dominate P contain s_1. Note that this means that there is a set D with $|D| \leq 6$ that dominates P and does not contain s_2, since we assumed that not all of such sets contain both s_1 and s_2. In this case, the vertices in $Q_1 \cap N_G(s_1)$ all have exclusively edges to vertices that do not have an edge to s_1 but do have an edge to s_2, since otherwise the edge would be redundant or $\{s_1, s_2\}$ would not dominate the vertex. See Figure 3b for an example of the vertices involved in such a structure. We can find that $|N_G(s_2) \cap P| \geq |Q_1 \cap N_G(s_1)|$. This follows from the fact that each vertex in $Q_1 \cap N_G(s_1)$ has at least two edges to vertices in $N_G(s_2) \cap P$ and that each vertex $v \in N_G(s_2) \cap P$ can have an edge to at most two vertices in $Q_1 \cap N_G(s_1)$. If v has edges to three vertices in $Q_1 \cap N_G(s_1)$, then v, s_1, s_2 and those three vertices would form $K_{3,3}$ when we contract the edges between s_2 and $N_G(s_2) \setminus \{v\}$. In a similar vein, any vertex in D can have an edge to at most two of these vertices in $N_G(Q_1 \cap N_G(s_1)) \setminus \{s_1\}$ due to planarity and thus can dominate at most three of them. However, there are more than $6 \cdot 3 = 18$ of such vertices which contradicts D dominating P.

In both cases we reach a contradiction, thus we must have that $|Q_1 \cap N_G(s_1)| \leq 6 \cdot 73$. The same bound holds for the vertices of Q_1 that are adjacent to s_2 through analogous reasoning, and since $Q_1 \subseteq N_G(s_1) \cup N_G(s_2)$, we must have that $|Q_1| \leq 2 \cdot 6 \cdot 73 = \mathcal{O}(1)$. $\lhd$

Summarizing so far, we have shown that $|P_1| = \mathcal{O}(1)$ and $|P_{\geq 3}| = |Q_1| + |Q_2| + |Q_{\geq 3}| = \mathcal{O}(1)$. This only leaves P_2, the vertices of degree two. All such vertices must have at least one edge to $N_G(P)$ since it dominates P. There are two options for the other edge. It could be an edge to another vertex in P_2 which in turn has an edge to some vertex in $N_G(P)$. Let P_2^* be the set of P_2 vertices involved in such a structure. If there were seven or more of such structures between a pair of vertices in $N_G(P)$, then the edge between the degree two vertices would have been redundant.

Thus, since $|N_G(P)| = \mathcal{O}(1)$, also $|P_2^*| = \mathcal{O}(1)$. Finally, consider the vertices of $P_2 \setminus P_2^*$. These are vertices of P of degree two with both neighbors in the set $N_G(P) \cup P_{\geq 3}$. The vertices of $P_2 \setminus P_2^*$ can be partitioned into $\mathcal{O}(1)$ wide diamonds, namely the wide diamonds of the form $N_G(x) \cap N_G(y) \cap P_2$ for $x, y \in N_G(P) \cup P_{\geq 3}$. Since $|N_G(P) \cup P_{\geq 3}| = \mathcal{O}(1)$, we also end up with $\mathcal{O}(1)$ wide diamonds. We can use Lemma 10 for each wide diamond to get a replacement P' and k' that satisfy the requirements stated in the lemma. □

Having proven that we can reduce the size of these protrusions, we proceed similarly to as for counting feedback vertex sets. We argue that there is an algorithm that solves #minDS in Lemma 12 and combine all our results in Theorem 6.

Lemma 12. *There exists an algorithm that, given a planar graph G and integer k, computes* $\#\mathrm{minDS}(G,k)$ *in* $2^{\mathcal{O}(\sqrt{k})} \cdot \mathrm{poly}(n)$ *time.*

Proof: There exists an algorithm that solves the decision version of Dominating Set in $4^{\mathrm{tw}(G)} \cdot \mathrm{tw}(G)^{\mathcal{O}(1)} \cdot n$ time [12, Theorem 7.9]. This algorithm can straightforwardly be adapted to also count the number of minimum dominating sets without increasing the running time; see e.g. [36]. Furthermore, note that if G has a dominating set of size at most k, then $\mathrm{tw}(G) = \mathcal{O}(\sqrt{k})$ [15, Lemma 15.7]. As a final ingredient, we use that there exists a polynomial-time exact algorithm for branchwidth on planar graphs [30], and that any branch decomposition of width k can be turned into a tree decomposition of width $\mathcal{O}(k)$ in polynomial time [18]. Putting this together results in an algorithm that can compute $\#\mathrm{minDS}(G,k)$ in the mentioned time. □

Theorem 6. *There is a polynomial-time algorithm that, given a planar graph G and integer k, either*

- *outputs* $\#\mathrm{minDS}(G,k)$*, or*
- *outputs a planar graph G' and integer k' such that* $\#\mathrm{minDS}(G,k) = \#\mathrm{minDS}(G',k')$ *and* $|V(G')| = \mathcal{O}(k^3)$ *and* $k' = \mathcal{O}(k^3)$*.*

Proof: First, we can use the algorithm from Lemma 9 on G and k. If it reports that the domination number of G is greater than k, then $\#\mathrm{minDS}(G,k) = 0$. Else we find a 2-approximate dominating set S and a set Z as stated in the lemma. If there is a protrusion of G with respect to Z that consists of more than 2^k vertices, we know that $n > 2^k$ so we can run the algorithm from Lemma 12 to compute $\#\mathrm{minDS}(G,k)$ in $\mathrm{poly}(n)$ time. Otherwise, we know that each protrusion consists of at most 2^k vertices. Applying Lemma 11 to all of the $\mathcal{O}(k)$ protrusions gives us a graph G' with $\mathcal{O}(k) + \mathcal{O}(k) \cdot \mathcal{O}(k^2) = \mathcal{O}(k^3)$ vertices and an integer $k' = k + \mathcal{O}(k) \cdot \mathcal{O}(k^2) = \mathcal{O}(k^3)$ such that $\#\mathrm{minDS}(G,k) = \#\mathrm{minDS}(G',k')$. □

5 Conclusion

We introduced a new model of kernelization for counting problems: a polynomial-time preprocessing algorithm that either outputs the desired count, or reduces to a provably small instance with the same answer. We showed that for counting the number of minimum solutions of size at most k, a reduction to a graph of size $\mathrm{poly}(k)$ exists for two classic problems.

We believe that the new viewpoint on counting kernelization facilitates a general theory that can be explored for many problems beyond the ones considered here. By following the textbook proof [12, Lemma 2.2] that a decidable parameterized decision problem is fixed-parameter tractable if and only if it admits a kernel (of potentially exponential size), it is easy to show the following equivalence between fixed-parameter tractability of counting problems and our notion of counting kernelization.

Lemma 13. *Let $\mathcal{P}\colon \Sigma^* \times \mathbb{N} \to \mathbb{N}$ be a computable function for some finite alphabet Σ. Then the following two statements are equivalent:*

1. *There is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and an algorithm that, given an input $(x,k) \in \Sigma^* \times \mathbb{N}$, outputs $\mathcal{P}(x,k)$ in time $f(k) \cdot |x|^{\mathcal{O}(1)}$.*
2. *There is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and a polynomial-time algorithm that, given $(x,k) \in \Sigma^* \times \mathbb{N}$, either:*
   (a) outputs $\mathcal{P}(x,k)$, or
   (b) outputs $(x',k') \in \Sigma^* \times \mathbb{N}$ satisfying $|x'|, k' \leq f(k)$ and $\mathcal{P}(x,k) = \mathcal{P}(x',k')$.

Hence our view of counting kernelization is generic enough to capture all fixed-parameter tractable counting problems. Determining which counting problems have a *polynomial-size* kernel remains an interesting challenge.

In this work, we focused on counting *minimum-size* solutions (of size at most k). Apart from being of practical interest in several applications, this facilitates several steps in the design and analysis of our preprocessing algorithms. At present, we do not know whether the two considered problems have polynomial-size kernels when counting the number of inclusion-minimal solutions of size exactly k, or the number of (not necessarily minimal or minimum) solutions of size exactly k. We do not see a reason why it would be impossible to obtain polynomial-size counting kernelizations for these variants, but we have not been able to find gadgets that preserve the answer. We leave their investigation to future work.

For both problems we investigated, our preprocessing step effectively consists of reducing to an equivalent instance composed of a small core along with poly(k) simply structured parts, followed by replacing each such part by a small problem-specific gadget. In the world of decision problems, the theory of protrusion replacement [7, 16] gives a generic way of replacing such simply-structured parts by gadgets. Similarly, the condenser-extractor framework [22, 31] can be applied to generic problems as long as they can be captured in a certain type of logic. This leads to the question of whether, in our model of counting kernelization, the design of the gadgets can be automated. Can a notion of meta-kernelization be developed for counting problems?

To illustrate the obstacles that need to be overcome, we compare to the setting of meta-kernelization for decision problems [7, 16]. There, the approach typically consists of two steps. First, one shows that for inputs asking for a solution of size k, in polynomial time one can either deduce that there are no solutions of size k or compute an *r-protrusion decomposition* of the input graph for some constant r. It effectively decomposes the graph into a "central" part G_0 of size $\mathcal{O}(k)$, together with $\mathcal{O}(k)$ protrusions $G_1, \ldots, G_{\mathcal{O}(k)}$. Each protrusion G_i is an induced subgraph of treewidth at most r, whose open neighborhood is contained in G_0 and has size at most r. This implies that each protrusion can be analyzed efficiently (since its treewidth is small) and interacts with the rest of the graph through a constant number of vertices.

The second step in the standard approach to meta-kernelization then argues that each protrusion can be replaced by a constant-size gadget that has the same effect on global solutions. For problems with *finite integer index* [7, §2.3], the existence of such a gadget is deduced from the fact that the influence of a protrusion on global solutions can be characterized by a *signature* that consists of $f(r)$ integers (one for each possible state in which the boundary vertices may end up when choosing a partial solution within the protrusion) in the range of 0 to $g(r)$. These integers effectively encode "how much does the size of a solution inside the protrusion go up, when requiring that it leaves the boundary vertices in a certain state?". For many problems, this cost increase can be bounded in terms of the size of the boundary because when picking all boundary vertices in the solution, one may then use the cheapest-possible solution in the rest of the protrusion. One can therefore characterize the behavior of a protrusion by a sequence of $f(r)$ integers of magnitude $g(r)$ each, for some functions f and g in the boundary size r alone. This implies that the number of distinct sequences (and therefore the number of distinct behaviors for protrusions) is bounded by a function of r. Using the fact that a protrusion is a bounded-treewidth subgraph, one can efficiently compute the *smallest* protrusion with the same behavior, whose size depends only on r and is therefore constant. Replacing each of the protrusions $G_1, \ldots, G_{\mathcal{O}(k)}$ by a constant-size gadget then yields an equivalent instance on $\mathcal{O}(k)$ vertices.

Based on this view of meta-kernelization for decision problems, we can now highlight the difficulties for the counting version. When it comes to the decomposition step, we expect that an r-protrusion decomposition can be obtained for counting problems without difficulties. Indeed, we mimicked existing decompositions in our results for Feedback Vertex Set and Dominating Set in planar graphs. The difficulty lies in obtaining a suitable gadget by which to replace a protrusion. When counting minimum solutions, characterizing the behavior of a protrusion requires the signature to track, for each of the $f(r)$ possible states of the boundary, how many distinct optimal solutions within the protrusion realize that given state on the boundary. The difficulty is that these numbers cannot be bounded in terms of the boundary size r alone. A priori these numbers can be as large as n^k. Using the existence of a single-exponential FPT algorithm in a similar fashion as in our paper, one can typically ensure $\log n \leq k$, so that one can encode a single integer of magnitude n^k in $\log(n^k) \leq k^2$ bits, but even the latter is problematic for a generic argument. When the behavior of a protrusion with respect to counting solutions is characterized by a sequence of $f(r)$ integers whose magnitude depends on k, then the number of distinct behaviors is no longer bounded in terms of the boundary size r alone. This means that "the smallest protrusion with the same behavior" is no longer necessarily of constant size; its size can depend on k, possibly superpolynomially. That is where the existing approach to meta-kernelization breaks in the counting setting. For the two problems treated in our paper, we managed to overcome this difficulty by arguing that it suffices to reduce only protrusions of a very specific form (chains or wide diamonds, respectively) and by hand-crafting a gadget in which up to 2^k different solutions can be generated by only $\mathcal{O}(k)$ different vertices. In general, it is not clear that a protrusion in which superpolynomially many distinct solutions exist, can always be represented by a gadget on $k^{\mathcal{O}(1)}$ vertices that generate the same number of possibilities. Determining when and how this is possible is the main obstacle for generalizing our results.

These issues in developing protrusion replacement for counting problems bear some resemblance to the difficulty of doing protrusion replacement for *weighted* graphs: the presence of vertices of different weights also means that to characterize the behavior of each of the $f(r)$ states of the boundary vertices of a protrusion, one may need an integer whose magnitude cannot be bounded in terms of r. We refer to recent work by Włodarczyk [38] for further discussion on the difficulties of weighted protrusion replacement. It would be interesting to see whether these difficulties can be overcome.

References

[1] J. Alber, M. R. Fellows, and R. Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004. doi:10.1145/990308.990309.

[2] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

[3] 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 J. Comput.*, 27(4):942–959, 1998. doi:10.1137/S0097539796305109.

[4] Bart M. P. Jansen. The power of preprocessing: Gems in kernelization. https://www.win.tue.nl/~bjansen/talks/BonnGemsInKernelization.pptx, 2016.

[5] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In S. R. Kosaraju, D. S. Johnson, and A. Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, STOC 1993*, pages 226–234. ACM, 1993. doi:10.1145/167088.167161.

[6] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

[7] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) kernelization. *J. ACM*, 63(5):44:1–44:69, 2016. doi:10.1145/2973749.

[8] J. F. Buss and J. Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993. doi:10.1137/0222038.

[9] Y. Cao, J. Chen, and Y. Liu. On feedback vertex set: New measure and new structures. *Algorithmica*, 73(1):63–86, 2015. doi:10.1007/s00453-014-9904-6.

[10] R. Curticapean. The simple, little and slow things count: on parameterized counting complexity. *Bull. EATCS*, 120, 2016. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/445.

[11] R. Curticapean, H. Dell, and D. Marx. Homomorphisms are a good basis for counting small subgraphs. In H. Hatami, P. McKenzie, and V. King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 210–223. ACM, 2017. doi:10.1145/3055399.3055502.

[12] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms.* Springer, 2015. doi:10.1007/978-3-319-21275-3.

[13] H. Fernau. *Parameterized algorithmics: A graph-theoretic approach.* PhD thesis, Habilitationsschrift, Universität Tübingen, Germany, 2005.

[14] J. Flum and M. Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.

[15] F. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: theory of parameterized preprocessing.* Cambridge University Press, 2019. doi:10.1017/9781107415157.

[16] F. V. Fomin. Protrusions in graphs and their applications. In V. Raman and S. Saurabh, editors, *Parameterized and Exact Computation - 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, volume 6478 of *Lecture Notes in Computer Science*, page 3. Springer, 2010. doi:10.1007/978-3-642-17493-3_2.

[17] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. *SIAM J. Comput.*, 49(6):1397–1422, 2020. doi:10.1137/16M1080264.

[18] I. V. Hicks, A. M. Koster, and E. Kolotoğlu. Branch and tree decomposition techniques for discrete optimization. In *Emerging Theory, Methods, and Applications*, pages 1–29. INFORMS, 2005. doi:10.1287/educ.1053.0017.

[19] Y. Iwata. Linear-time kernelization for feedback vertex set. In I. Chatzigiannakis, P. Indyk, F. Kuhn, and A. Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 68:1–68:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.68.

[20] B. M. P. Jansen and B. van der Steenhoven. Kernelization for counting problems on graphs: Preserving the number of minimum solutions. In *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6-8, 2023, Amsterdam, The Netherlands*, volume 285 of *LIPIcs*, pages 27:1–27:15, 2023. doi:10.4230/LIPICS.IPEC.2023.27.

[21] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993. doi:10.1137/0222066.

[22] E. J. Kim, M. J. Serna, and D. M. Thilikos. Data-compression for parametrized counting problems on sparse graphs. In W. Hsu, D. Lee, and C. Liao, editors, *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, volume 123 of *LIPIcs*, pages 20:1–20:13, 2018. doi:10.4230/LIPIcs.ISAAC.2018.20.

[23] D. Lokshtanov, P. Misra, S. Saurabh, and M. Zehavi. Kernelization of counting problems. In V. Guruswami, editor, *Proceedings of the 15th Innovations in Theoretical Computer Science Conference, ITCS 2024*, volume 287 of *LIPIcs*, pages 77:1–77:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ITCS.2024.77.

[24] M. Luby and E. Vigoda. Fast convergence of the glauber dynamics for sampling independent sets. *Random Struct. Algorithms*, 15(3-4):229–241, 1999.

[25] C. McCartin. Parameterized counting problems. *Annals of Pure and Applied Logic*, 138(1):147–182, 2006. doi:10.1016/j.apal.2005.06.010.

[26] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science (New York, N.Y.)*, 298:824–7, 11 2002. doi:10.1126/science.298.5594.824.

[27] N. Nishimura, P. Ragde, and D. M. Thilikos. Parameterized counting algorithms for general graph covering problems. In F. K. H. A. Dehne, A. López-Ortiz, and J. Sack, editors, *Algorithms and Data Structures, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, volume 3608 of *Lecture Notes in Computer Science*, pages 99–109. Springer, 2005. doi:10.1007/11534273_10.

[28] P. Orponen. Dempster's rule of combination is #P-complete. *Artif. Intell.*, 44(1-2):245–253, 1990. doi:10.1016/0004-3702(90)90103-7.

[29] D. Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1-2):273–302, 1996. doi:10.1016/0004-3702(94)00092-1.

[30] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Comb.*, 14(2):217–241, 1994. doi:10.1007/BF01215352.

[31] D. M. Thilikos. Compactors for parameterized counting problems. *Comput. Sci. Rev.*, 39:100344, 2021. doi:10.1016/j.cosrev.2020.100344.

[32] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, 2010. doi:10.1145/1721837.1721848.

[33] M. Thurley. Kernelizations for parameterized counting problems. In J. Cai, S. B. Cooper, and H. Zhu, editors, *Theory and Applications of Models of Computation, 4th International Conference, TAMC 2007, Shanghai, China, May 22-25, 2007, Proceedings*, volume 4484 of *Lecture Notes in Computer Science*, pages 703–714. Springer, 2007. doi:10.1007/978-3-540-72504-6_64.

[34] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. doi:10.1137/0220053.

[35] L. G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

[36] J. M. M. van Rooij. Fast algorithms for join operations on tree decompositions. In F. V. Fomin, S. Kratsch, and E. J. van Leeuwen, editors, *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 262–297. Springer, 2020. doi:10.1007/978-3-030-42071-0_18.

[37] D. J. A. Welsh. Graph theory and theoretical physics. *The Mathematical Gazette*, 54(390):432–433, 1970. doi:10.2307/3613919.

[38] M. Wlodarczyk. Losing treewidth in the presence of weights. In Y. Azar and D. Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025*, pages 3743–3761. SIAM, 2025. doi:10.1137/1.9781611978322.125.

A An FPT Algorithm for #minFVS

Lemma 8. *There is an algorithm that, given a graph G and integer k, computes* $\#\mathrm{minFVS}(G, k)$ *in* $2^{\mathcal{O}(k)} \cdot \mathrm{poly}(n)$ *time.*

Proof: The pseudocode of an algorithm to solve the disjoint minimum feedback vertex set counting problem is given in Algorithm 2 and is based on the algorithm by Cao, Chen, and Liu [9]. In fact, our algorithm solves a slightly more general version of the problem, where the vertices of the graph G are weighted by a function $w\colon V(G) \to \mathbb{N}$. In case G has no FVS of size at most k that is disjoint from W, then $\#\text{DJ-FVS}(G, w, W, k)$ will output a pair (a, b) with $a = \infty$ and $b = 0$. Otherwise, a will be the size of a minimum FVS disjoint of W and

$$b = \sum_{\substack{|S|=a,\\ S\in \mathrm{FVS}(G),\\ S\cap W=\emptyset}} \prod_{v\in S} w(v).$$

We refer to this value as the *weighted disjoint minimum FVS sum* of G. The weight of a vertex v essentially models the number of distinct alternatives there are for a vertex v that, in the context of choosing them for a minimum FVS of G, achieve the same result. When we assign a weight of one to each vertex of a graph G, then the weighted minimum FVS sum of G is equal to the number of minimum feedback vertex sets of G.

In the pseudocode we make use of binary operator $\oplus$, which is defined to work on pairs as follows:

$$(a_1, b_1) \oplus (a_2, b_2) = \begin{cases} (a_1, b_1) & \text{if } a_1 < a_2 \\ (a_2, b_2) & \text{if } a_1 > a_2 \\ (a_1, b_1 + b_2) & \text{if } a_1 = a_2 \end{cases}$$

For the operators $+$ and $\cdot$ we assume element-wise functionality when applied to pairs, i.e. $(a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2)$. Furthermore, for a weight function w of G and $X \subseteq V(G)$, we use $w|_X$ to denote the restriction of w to X.

We shall now explain how the #DJ-FVS algorithm works by going over the pseudocode in Algorithm 2. It makes use of a branching strategy with measure function $\ell + k$, where ℓ is the number of connected components of $G[W]$.

Lines 1-3 are the base cases of the algorithm. In lines 4-5 we remove vertices of degree at most one, which corresponds to (R2). In lines 6-7 we contract the chains of G existing in $G - W$ one edge at a time. Note for line 8 that H is a forest since W is an FVS of G. For lines 9-10, if a vertex v would form a cycle with W it should be contained in all solutions so we recurse on this choice. In lines 11-14, vertex v has at least two neighbors in W, but since $G[W \cup \{v\}]$ does not form a cycle these neighbors must be in different connected components of $G[W]$. Thus we branch on v not being in a solution, which corresponds with adding v to W, and on v being in a solution by removing it from the graph. In the former branch ℓ decreases, while in the latter k decreases.

In line 15, we choose a vertex $v \in V(H)$ that is not a leaf of tree H such that at most one of its neighbors in H is not a leaf of H. Note that such a vertex always exists for a tree that does not consist of only leaves. Furthermore, at this point of the algorithm no tree of H can consist of only leaves. If a tree of H consists of a single leaf, then either it has degree one in G, but then lines 4-5 would have gotten rid of it, or it has degree at least two, in which case the if condition on line 9 or the if condition on line 11 would have been satisfied. If a tree of H consists of two leaves, then either both have degree two in which case the edge between them would have been contracted by lines 6-7, or one of the if statements of line 9 or line 11 would have been applicable to one of the two leaves.

First consider the case where v has one neighbor in W (lines 16-22). In that case we pick c in line 16 to be a child of v. For these two vertices, we branch over all possible combinations of whether or not they should be in a solution, while realizing that a minimum FVS that contains v can never contain c. Note that if $G[W \cup \{v, c\}]$ does not form a cycle, both v and c must have a neighbor in a different connected component of W so the branch in line 19 decreases ℓ. The branches on line 20 and 21 decrease k while not increasing ℓ.

Finally, we have the case that v has no neighbors in W (lines 23-30). That must mean that v has at least two children, which are all leaves by how we chose v. If v would have had only one child, then the edge between v and the child would have been contracted in lines 6-7. Thus we let c_1 and c_2 be two distinct children of v. Again, we branch over all viable combinations of how these three vertices can be part of minimum solutions. The logic here is similar to that of the previous case. Here as well, in all branches, the measure $k + \ell$ decreases.

Since the algorithm branches in at most four directions and the time per iteration is polynomial in n, the runtime of the disjoint algorithm becomes $4^{k+\ell} \cdot n^{O(1)}$. We can use Algorithm 2 to compute #minFVS(G, k) by taking an FVS Z of G and running the disjoint algorithm for all subsets of Z, simulating the ways an FVS can intersects Z. The pseudocode for this compression algorithm can be seen in Algorithm 1. To get an FVS of G of size at most k, we can simply run one of the existing algorithms designed for this, for example the one by Cao, Chen, and Liu [9] which runs in time $\mathcal{O}(3.83^k) \cdot \text{poly}(n)$. If this reports that no such FVS exists, we output #minFVS$(G, k) = 0$. Otherwise, we use the found FVS for the compression algorithm. Using the fact that the FVS used by the compression algorithm is of size at most k, we can bound the runtime of the complete algorithm to compute #minFVS(G, k) at $17^k \cdot n^{O(1)}$. □

Algorithm 1 #FVS-COMPRESSION(G, k, Z)

```
Input: Graph $G$, integer $k$, FVS $Z$ of $G$ of size at most $k$
Output: A pair $(a, b)$ with $a$ being the feedback vertex number of $G$ and
$b = \#\text{minFVS}(G, k)$
1: $s = (\infty, 0)$
2: for $X_Z \subseteq Z$ do
3:     $s' = (|X_Z|, 0) + \#\text{DJ-FVS}(G - X_Z,\ w,\ Z \setminus X_Z,\ k - |X_Z|)$ with $\forall v \in V(G - X_Z) : w(v) = 1$
4:     $s = s \oplus s'$
5: return s
```

Algorithm 2 #DJ-FVS(G, w, W, k)

```
Input: Graph $G$, weight function $w\colon V(G) \to \mathbb{N}$, FVS $W$ of $G$, integer $k$
Parameter: $k + \ell$, with $\ell = \#$ components of $G[W]$
Output: A pair $(a, b)$ where $a$ is the size of a minimum FVS $S$ of $G$ such that
$S \cap W = \emptyset$ and $b$ is the weighted disjoint minimum FVS sum of $G$. If no such FVS
of size at most $k$ exists, then $(a, b) = (\infty, 0)$.
1: if $k < 0$ then return $(\infty, 0)$
2: if $G[W]$ has a cycle then return $(\infty, 0)$
3: if $G - W$ is empty then return $(0, 1)$
4: if $\exists v \in V(G - W)$: $\deg_G(v) \leq 1$ then
5:     return #DJ-FVS($G - v$, $w|_{V(G)\setminus\{v\}}$ ,$W$,$k$)
6: if $\exists \{v, u\} \in E(G)$ : $\deg_G(v) = \deg_G(u) = 2$ and $v, u \notin W$ then
7:     return #DJ-FVS($G'$, $w'$ $W$, $k$), where $G'$ is $G$ with edge $\{v, u\}$ contracted to a single
       vertex $s$ and $w'$ is the weight function $w|_{V(G')}$ with $w'(s) = w(v) + w(u)$.
8: Let $H$ be forest $G - W$.
9: if $\exists v \in V(H)$: $G[W \cup \{v\}]$ has a cycle then
10:     return $(1, 0) + (1, w(v))\cdot$ #DJ-FVS($G - v$, $w|_{V(G)\setminus\{v\}}$, $W$, $k - 1$)
11: if $\exists v \in V(H)$: $|N_G(v) \cap W| \geq 2$ then
12:     $X_0$ =#DJ-FVS($G$, $w$, $W \cup \{v\}$, $k$)
13:     $X_1 = (1, 0) + (1, w(v))\cdot$ #DJ-FVS($G - v$, $w|_{V(G)\setminus\{v\}}$, $W$, $k - 1$)
14:     return $X_0 \oplus X_1$
15: Let $v \in V(H)$ be a vertex that is not a leaf of $H$ such that at most one vertex in $N_H(v)$ is not
    a leaf of $H$.
16: if $|N_G(v) \cap W| = 1$ then
17:     Pick $c \in V(H)$ such that $N_G(c) = \{v, x\}$ for some $x \in W$
18:     if $G[W \cup \{v, c\}]$ forms a cycle then $X_{00} = (\infty, 0)$
19:     else $X_{00}$ = #DJ-FVS($G$, $w$, $W \cup \{v, c\}$, $k$)
20:     $X_{10} = (1, 0) + (1, w(v))\cdot$ #DJ-FVS($G - v$, $w|_{V(G)\setminus\{v\}}$, $W$, $k - 1$)
21:     $X_{01} = (1, 0) + (1, w(c))\cdot$ #DJ-FVS($G - c$, $w|_{V(G)\setminus\{c\}}$, $W \cup \{v\}$, $k - 1$)
22:     return $X_{00} \oplus X_{10} \oplus X_{01}$
23: if $|N_G(v) \cap W| = 0$ then
24:     Pick $c_1, c_2 \in V(H), c_1 \neq c_2$ such that $N_G(c_1) = \{v, x\}$ and $N_G(c_2) = \{v, y\}$ for some
        $x, y \in W$
25:     if $G[W \cup \{v, c_1, c_2\}]$ forms a cycle then $X_{000} = (\infty, 0)$
26:     else $X_{000}$ = #DJ-FVS($G$, $w$, $W \cup \{v, c_1, c_2\}$, $k$)
27:     $X_{100} = (1, 0) + (1, w(v))\cdot$ #DJ-FVS($G - v$, $w|_{V(G)\setminus\{v\}}$, $W$, $k - 1$)
28:     $X_{010} = (1, 0) + (1, w(c_1))\cdot$ #DJ-FVS($G - c_1$, $w|_{V(G)\setminus\{c_1\}}$, $W \cup \{v, c_2\}$, $k - 1$)
29:     $X_{001} = (1, 0) + (1, w(c_2))\cdot$ #DJ-FVS($G - c_2$, $w|_{V(G)\setminus\{c_2\}}$, $W \cup \{v, c_1\}$, $k - 1$)
30:     return $X_{000} \oplus X_{100} \oplus X_{010} \oplus X_{001}$
```