Sidestepping Barriers for Dominating Set in Parameterized Complexity

Ioannis Koutis
New Jersey Institute of Technology, NJ, USA

Michał Włodarczyk
University of Warsaw, Poland

Meirav Zehavi
Ben-Gurion University of the Negev, Beerhseba, Israel

Abstract

We study the classic Dominating Set problem with respect to several prominent parameters. Specifically, we present algorithmic results that sidestep time complexity barriers by the incorporation of either approximation or larger parameterization. Our results span several parameterization regimes, including: (i,ii,iii) time/ratio-tradeoff for the parameters *treewidth*, *vertex modulator to constant treewidth* and *solution size*; (iv,v) FPT-algorithms for the parameters *vertex cover number* and *feedback edge set number*; and (vi) compression for the parameter *feedback edge set number*.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Dominating Set, Parameterized Complexity, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.IPEC.2023.31

Related Version *Full Version*: `https://arxiv.org/abs/2309.15645`

Funding Supported by BGU–NJIT Joint Seed Research Fund and ERC Starting Grant titled PARAPATH.

1 Introduction

The Dominating Set problem is one of the most central problems in Parameterized Complexity [8, 4]. The input to Dominating Set consists of an n-vertex graph G, and the objective is to output a minimum-size subset $U \subseteq V(G)$ that is a *dominating set* – that is, the closed neighborhood of U in G equals $V(G)$, or, in other words, every vertex in $V(G) \setminus U$ is adjacent in G to at least one vertex in U. When parameterized by the solution size and stated as a decision problem, the input also consists of a non-negative integer k, and the objective is to determine whether there exists a subset $U \subseteq V(G)$ of size at most k that is a dominating set.

From the perspective of Parameterized Complexity, Dominating Set parameterized by the sought solution size k is very hard. First, Dominating Set is W[2]-complete [8] (and, clearly, in XP).[1] In fact, Dominating Set and Set Cover are the two most well-studied W[2]-hard problems in Parameterized Complexity. Moreover, under the Strong Exponential Time Hypothesis (SETH), Dominating Set cannot be solved in $f(k) \cdot n^{k-\epsilon}$ time [19]. Still, for every integer $k \geq 7$, Dominating Set is solvable in $n^{k+o(1)}$ time [9].

[1] We refer to Section 2 for notations and concepts not defined in the introduction.

From the perspective of approximation (and parameterized approximation), Dominating Set is also very hard. Unless P=NP, Dominating Set does not admit a polynomial-time $(1-\epsilon)\ln n$-approximation algorithm for any fixed $\epsilon > 0$ [7] (see also [10]). However, it admits a polynomial-time $(\ln n - \ln\ln n + O(1))$-approximation algorithm [23]. Moreover, under the SETH, Dominating Set does not even admit a $g(k)$-approximation $f(k)\cdot n^{k-\epsilon}$-time algorithm for any computable functions g and f of k and fixed $\epsilon > 0$ [20]. (Observe that this statement generalizes the one above by [19].) Similar (but weaker) results of hardness of approximation in the setting of Parameterized Complexity also exist under other assumptions, including the ETH, W[1]≠FPT, and the k-SUM hypothesis [20].

Concerning structural parameters, the most well-studied parameters in Parameterized Complexity are treewidth and vertex cover number [8, 4]. Regarding Dominating Set, on the positive side, the problem is easily solvable in $O(3^{\mathsf{tw}}\cdot n)$ time [21]. Here, tw is the treewidth of the input graph. However, under the SETH, Dominating Set cannot be solved in $(3-\epsilon)^{\mathsf{tw}}\cdot n^{O(1)}$ time for any fixed $\epsilon > 0$ [16]. Similarly to the case of the parameter solution size k, we again have (essentially) matching upper and lower bounds in terms of time complexity. Moreover, it is not hard to see that, under the any of the SETH and the Set Cover Conjecture, Dominating Set cannot be solved in $(2-\epsilon)^{\mathsf{vc}}\cdot n^{O(1)}$ time for any fixed $\epsilon > 0$ (see Section 5). Here, vc is the vertex cover number of the input graph.

Lastly, we note that the weighted version of Dominating Set is, similarly, approximable in polynomial time within factor $O(\log n)$, and solvable in $O(3^{\mathsf{tw}}\cdot n)$ time. Further, being more general, all negative results carry to it as well.

1.1 Our Contribution

Our contribution is fivefold, concerning five different parameterizations.

I. Treewidth. First, in Section 3, we consider the treewidth tw of the given graph as the parameter. We prove that Weighted Dominating Set admits a 2-approximation $O(\sqrt{6}^{\mathsf{tw}}\cdot\mathsf{tw}^{O(1)}\cdot n)$-time algorithm. Our proof is based on "decoupling" the task of domination of the entire input graph G into two separate tasks: we compute a partition (V_1, V_2) of G based on a proposition of [17], and then consider the domination of each V_i separately. We remark that the way that we use the aforementioned proposition is very different than the way it is originally used in [17]. Here, we remind that under the SETH, Dominating Set cannot be solved in $(3-\epsilon)^{\mathsf{tw}}\cdot n^{O(1)}$ time for any fixed $\epsilon > 0$.

II. Modulator to Constant Treewidth. Second, in Section 4, we consider the parameter tw_d, the minimum-size of a vertex modulator of the given graph to treewidth d, for any fixed $d \in \mathbb{N}$. We note that, for any graph G, $\mathsf{tw} \leq \mathsf{tw}_d + d$. We prove that Weighted Dominating Set admits a 2-approximation $O(2^{\mathsf{tw}_d}\cdot n)$-time algorithm. As before, our proof is based on "decoupling" the task of domination of the entire input graph G into two separate tasks: now, these are the task of the domination of the modulator, and the task of the domination of the rest of G. Unlike before, the resolution of these two tasks is different. Concerning the tightness of our result, we refer the reader to Conjecture 30, where we conjecture that the same time complexity cannot be attained by an exact algorithm.

III. Vertex Cover Number. Third, in Section 5, we consider the vertex cover number vc of the given graph as the parameter. We prove that Weighted Dominating Set admits an $O(2^{\mathsf{vc}}\cdot n)$-time algorithm. Our proof is partially based on the idea of our algorithm for the parameter tw_d, combined with the observation that some vertices in the independent set (being the complement of the given vertex cover) are "forced" to be picked, after having chosen which vertices to pick from the vertex cover. From the perspective of impossibility results, we observe that under any of the SETH and the Set Cover Conjecture, Dominating Set cannot be solved in $(2-\epsilon)^{\mathsf{vc}} \cdot n^{O(1)}$ time for any fixed $\epsilon > 0$, thus our time complexity is tight.

IV. Feedback Edge Set Number. Fourth, in Section 6, we consider the parameter fes (feedback edge set number), the minimum-size of a set of edges whose removal transforms the input graph into a forest. Notice that this parameter is a relaxation of tw_d (for any fixed $d \in \mathbb{N}$), which, in turn, is a relaxation of tw. We present two theorems. The first theorem is that Dominating Set admits an $O(3^{\frac{\mathsf{fes}}{2}} \cdot n)$-time algorithm. To this end, we prove the following combinatorial lemma, which is of independent interest: For any graph G, $\mathsf{tw}_2(G) \leq \frac{\mathsf{fes}(G)}{2}$. (Moreover, we prove that there exists an algorithm that, given a graph G, outputs a subset $M \subseteq V(G)$ such that $|M| \leq \frac{\mathsf{fes}(G)}{2}$ and $\mathsf{tw}(G-M) \leq 2$ in $O(\mathsf{fes}(G)+n)$ time.) The second theorem is that an instance G of the Dominating Set problem with $\mathsf{fes}(G) = k$ can be compressed in linear time into a "relaxed" instance of the problem on a graph $\hat{G}$ with $O(k)$ edges, requiring the minimum domination of a subset of the vertices in $\hat{G}$.

V. Solution Size. Fifth, we consider the solution size k as the parameter. We prove that, for any fixed $0 \leq \alpha < 1$, Dominating Set admits a $((1-\alpha)\ln n + O(1))$-approximation $n^{\alpha k + O(1)}$-time algorithm. The proof of this theorem is the simplest one in our article, based on the combination of an exhaustive search (to uncover part of the solution) and a known approximation algorithm. This approach somewhat resembles that of [5]. Here, we remind that (under plausible complexity-theoretic assumptions) it is unlikely for an exact algorithm to run in $n^{\alpha k + O(1)}$ time, or for a polynomial-time algorithm to have approximation factor $((1-\alpha)\ln n + O(1))$.

Due to space constraints, the compression result in Contribution IV and Contribution V are deferred to the full version of this paper.

1.2 Other Related Works

Here, we briefly survey a few works not already mentioned that are directly relevant or related to ours. First, we note that Dominating Set can be solved in linear time on series-parallel graphs [14]. In particular, the class of series-parallel graphs is a (strict) subclass of the class of graphs of treewidth 2.

The restriction of Dominating Set to planar graphs is known to be both solvable in $2^{O(\sqrt{k})}n$ time (thus, it is in FPT) and admit an EPTAS [12]. Similar results exist also for more general graph classes, such as H-minor free graphs [11] and graphs of bounded expansion [1]. Moreover, Dominating Set is also in FPT (specifically, it is solvable in $2^{O(k)}n^{O(1)}$ time, and admits a polynomial kernel) on claw-free graphs, but it remains W[2]-complete on $K_{1,t}$-free graphs for any $t \geq 4$ [6, 13]. Additionally, Dominating Set is W[1]-hard on unit disk graphs [18].

With respect to exact exponential-time algorithms, the currently best known running time upper bound is of $O(1.4969^n)$, based on the branch-and-reduce method [22].

2 Preliminaries

Given a function $f : U \to \mathbb{R}$ and a subset $U' \subseteq U$, let $f(U') = \sum_{u \in U'} f(u)$. When f is interpreted as a weight function, we will refer to $f(U')$ as the weight of U'.

Standard Graph Notation. Throughout the article, we deal with simple, finite, undirected graphs. Given a graph G, let $V(G)$ and $E(G)$ denote its vertex and edge sets, respectively. When no confusion arises, we denote $|V(G)| = n$ and $|E(G)| = m$. Given a vertex $v \in V(G)$, let $N_G(v) = \{u \in V(G) : \{u, v\} \in E(G)\}$ and $N_G[v] = N_G(v) \cup \{v\}$. Given a subset $U \subseteq V(G)$, let $N_G(U) = \bigcup_{u \in U} N_G(u) \setminus U$ and $N_G[U] = \bigcup_{u \in U} N_G[u]$. When G is clear from context, we drop it from the subscripts. Given a subset $U \subseteq V(G)$, let $G[U]$ denote the subgraph of G induced by U, and let $G - U$ denote the graph $G[V(G) \setminus U]$. Given a subset $F \subseteq E(G)$, let $G - F$ denote the graph on vertex set $V(G)$ and edge set $E(G) \setminus F$. Given subsets $A, B \subseteq V(G)$, we say that A *dominates* B if for every $b \in B$, $N[b] \cap A \neq \emptyset$. A *dominating set* of G is a subset $U \subseteq V(G)$ that dominates $V(G)$. A *vertex cover* of G is a subset $U \subseteq V(G)$ such that $G - U$ is edgeless (i.e., $E(G - U) = \emptyset$). Let $\mathsf{vc}(G)$ denote the minimum size of a vertex cover of G. A *feedback edge set* of G is a subset $F \subseteq E(G)$ such that $G - F$ is a forest. Let $\mathsf{fes}(G)$ denote the minimum size of a feedback edge set of G. We note that $\mathsf{vc}(G)$ and $\mathsf{fes}(G)$ are incomparable. When G is immaterial or clear from context, we denote $\mathsf{vc} = \mathsf{vc}(G)$ and $\mathsf{fes} = \mathsf{fes}(G)$. A *cactus* is a connected graph in which any two simple cycles have at most one vertex in common. We will slightly abuse this term: given graph G such that each connected component of G is a cactus, we will call G a cactus as well.

Problem Definitions. The Dominating Set problem is defined as follows: The input consists of an n-vertex graph G, and the objective is to output a minimum-size dominating set in G. The Weighted Dominating Set is defined similarly: Here, the input also consists of a weight function $w : V(G) \to \mathbb{N}$, and the objective is to output a minimum-weight dominating set in G. When parameterized by the solution size k, we consider the decision version of Dominating Set, and suppose that the input also consists of a non-negative integer k. Then, the objective is to determine whether G has a dominating set of size at most k. Parameterization by the solution size k can also be defined for Weighted Dominating Set. However, we find it to be somewhat less natural (particularly when considered from the perspective of parameterized approximation), and therefore we do not consider it in this paper. Still, we mention that, here, the objective would be to find a minimum-weight dominating set in G among all those of size at most k, if one exists.

The Set Cover problem is defined as follows: The input consists of a universe U and a family $\mathcal{F} \subseteq 2^U$ of subsets of U , and the objective is to output a minimum-size subfamily $\mathcal{S} \subseteq \mathcal{F}$ such that $U = \bigcup \mathcal{S}$. (Here, without loss of generality, we suppose that $U = \bigcup \mathcal{F}$, so there necessarily exist a solution.) The Weighted Set Cover problem is defined similarly: Here, the input also consists of a weight function $w : \mathcal{F} \to \mathbb{N}$, and the objective is to output a minimum-weight subfamily $\mathcal{S} \subseteq \mathcal{F}$ such that $U = \bigcup \mathcal{S}$. We require a slight generalization of this problem, called Generalized Weighted Set Cover, defined as follows: The input $(U, \mathcal{F}, w)$ is the same, and the objective is to output, for every subset $A \subseteq U$, a minimum-weight subfamily $\mathcal{S}_A \subseteq \mathcal{F}$ such that $U_A = \bigcup \mathcal{S}_A$.

Width Measures. The treewidth of a graph is a standard measure of its “closeness” to a tree, defined as follows.

▶ **Definition 1.** *A* tree decomposition *of a graph G is a pair $\mathcal{T} = (T, \beta)$, where T is a rooted tree and β is a function from $V(T)$ to $2^{V(G)}$, that satisfies the following conditions.*

- *For every edge $\{u, v\} \in E(G)$, there exists $x \in V(T)$ such that $\{u, v\} \subseteq \beta(x)$.*
- *For every vertex $v \in V(G)$, $T[\{x \in V(T) : v \in \beta(x)\}]$ is a tree on at least one vertex.*

The width *of (T, β) is $\max_{x \in V(T)} |\beta(x)| - 1$. The* treewidth *of G, denoted by $\mathsf{tw}(G)$, is the minimum width over all tree decompositions of G. For every $x \in V(T)$, $\beta(x)$ is called a* bag*, and $\gamma(x)$ denotes the union of the bags of x and the descendants of x in T.*

When G is immaterial or clear from context, we denote $\mathsf{tw} = \mathsf{tw}(G)$. Following the standard custom in parameterized algorithmics, when we consider a problem parameterized by tw, we suppose that we are given a tree decomposition $\mathcal{T}$ of width tw. Also, following the standard custom, we do not rely on a supposition that the width of $\mathcal{T}$ is tw in the sense that, if the width of $\mathcal{T}$ is larger, then our algorithmic result holds where tw is replaced by this width.

Given a graph G, a *path decomposition* of G is a tree decomposition (T, β) where T is a path, and the PATHWIDTH of G is the minimum width over all path decompositions of G.

For $d \in \mathbb{N} \cup \{0\}$ and a graph G, let $\mathsf{tw}_d(G)$ denote the minimum size of a vertex set whose deletion from G results in a graph of treewidth at most d. Observe that, for any graph G, $\mathsf{tw}_0(G) = \mathsf{vc}(G)$. When G is immaterial or clear from context, we denote $\mathsf{tw}_d = \mathsf{tw}_d(G)$. Following the standard custom in parameterized algorithmics, when we consider a problem parameterized by tw_d, we suppose that we are given a vertex set M of size tw_d whose deletion from the input graph results in a graph of treewidth at most d. Also, following the standard custom, we do not rely on a supposition that $|M| = \mathsf{tw}_d$ in the sense that, if $|M|$ is larger, then our algorithmic result holds where tw_d is replaced by $|M|$.

For the design of algorithms based dynamic programming, it is convenient to work with *nice* tree decompositions, defined as follows.

▶ **Definition 2.** *A tree decomposition (T, β) of a graph G is* nice *if for the root $r = \mathsf{root}(T)$ of T, $\beta(r) = \emptyset$, and each node $x \in V(T)$ is of one of the following types.*

- **Leaf***: x is a leaf in T and $\beta(x) = \emptyset$.*
- **Forget***: x has one child, y, and there is a vertex $v \in \beta(u)$ such that $\beta(x) = \beta(y) \setminus \{v\}$.*
- **Introduce***: x has one child, y, and there is a vertex $v \in \beta(x)$ such that $\beta(x) \setminus \{v\} = \beta(y)$.*
- **Join***: x has two children, y and z, and $\beta(x) = \beta(y) = \beta(z)$.*

▶ **Proposition 3** ([2])**.** *Given a tree decomposition (T, β) of a graph G, a nice tree decomposition of G of the same width as (T, β) can be constructed in linear-time (specifically, $O(\mathsf{w}^{O(1)} \cdot n)$ where w is the width of (T, β)).*

Due to Proposition 3, when we deal with nice tree decompositions of width tw, we suppose that $|V(T)| \leq O(\mathsf{tw}^{O(1)} \cdot n)$.

Parameterized Complexity. Let Π be an NP-hard problem. In the framework of Parameterized Complexity, each instance of Π is associated with a *parameter* k. Here, the goal is to confine the combinatorial explosion in the running time of an algorithm for Π to depend only on k. Formally, we say that Π is *fixed-parameter tractable (FPT)* if any instance (I, k) of Π is solvable in $f(k) \cdot |I|^{O(1)}$ time, where f is an arbitrary function of k. A weaker request is that for every fixed k, the problem Π would be solvable in polynomial time. Formally, we say that Π is *slice-wise polynomial (XP)* if any instance (I, k) of Π is solvable in $f(k) \cdot |I|^{g(k)}$ time, where f and g are arbitrary functions of k. Parameterized Complexity also provides methods to show that a problem is unlikely to be FPT. Here, the concept of W-hardness replaces the one of NP-hardness. For more information, we refer the reader to the book [4].

Essentially tight conditional lower bounds for the running times of parameterized algorithms often rely on the *Exponential-Time Hypothesis (ETH)*, the *Strong ETH (SETH)* and the *Set Cover Conjecture*. To formalize the statements of ETH and SETH, recall that given a formula φ in conjuctive normal form (CNF) with n variables and m clauses, the task of CNF-SAT is to decide whether there is a truth assignment to the variables that satisfies φ. In the p-CNF-SAT problem, each clause is restricted to have at most p literals. First, ETH asserts that 3-CNF-SAT cannot be solved in $O(2^{o(n)})$ time. Second, SETH asserts that for every fixed $\epsilon < 1$, there exists a (large) integer $p = p(\epsilon)$ such that p-CNF-SAT cannot be solved in $O((2-\epsilon)^n)$ time. Moreover, the Set Cover Conjecture states that for every fixed $\epsilon < 1$ Set Cover cannot be solved in $O((2-\epsilon)^n)$ time where n is the size of the universe.

Let P and Q be two parameterized problems. A *compression* (or *compression algorithm* for P is a polynomial-time procedure that, given an instance (x, k) of P, outputs an equivalent instance (x', k') of Q where $|x'|, k' \leq f(k)$ for some computable function f. Then, we say that P admits a *compression of size* $f(k)$. When $P = Q$, compression is called *kernelization*.

When a problem is parameterized by the solution size k, the concept of parameterized approximation must be clarified (given that we then deal with a decision problem). Here, the objective becomes the following where the sought approximation ratio is some α: If there exists a solution of size at most k, then we seek an α-approximate solution; else, we can output any solution. Of course, we do not know (as part of the input) which case is true.

3 Parameter: Treewidth

We will use a translation of (Weighted) Dominating Set into two instances of an easier problem, to attain a 2-approximation algorithm for (Weighted) Dominating Set.

▶ **Theorem 4.** *The* Weighted Dominating Set *problem parameterized by* tw *admits a* 2*-approximation* $O(\sqrt{6}^{\mathsf{tw}} \cdot \mathsf{tw}^{O(1)} \cdot n)$*-time algorithm.*

Towards the proof of this theorem, we first define the easier problem that we aim to solve.

▶ **Definition 5** (Half-Width Domination)**.** *In the* Weighted Half-Width Domination *problem, the input consists of a graph G, a vertex-weight function $w : V(G) \to \mathbb{N}$, a nice tree decomposition $\mathcal{T} = (T, \beta)$ of G of width* tw*, and a subset $D \subseteq V(G)$ such that for every $x \in V(T)$, $|\beta(x) \cap D| \leq \frac{\mathsf{tw}}{2} + O(1)$. The objective is to compute a subset $S \subseteq V(G)$ of minimum weight that dominates D.*

We would have liked to call the algorithm in the following proposition in order to directly solve the Weighted Half-Width Domination problem.

▶ **Proposition 6** ([4])**.** *The* Weighted Dominating Set *problem admits an* $O(3^{\mathsf{tw}} \cdot n)$*-time algorithm.*

Unfortunately, we cannot use it in a black-box manner – we need to modify the dynamic programming table used in the proof. Intuitively, we let vertices outside D correspond to two states (chosen, not chosen) instead of three (chosen, not chosen and dominated, not chosen and not dominated). This is done in the following lemma.

▶ **Lemma 7.** *The* Weighted Half-Width Domination *problem admits an* $O(\sqrt{6}^{\mathsf{tw}} \cdot \mathsf{tw}^{O(1)} \cdot n)$*-time algorithm.*

Proof. We first describe the algorithm. Let $(G, w, D, \mathcal{T} = (T, \beta)$ be the give input. We use dynamic programming, and start with the formal definition of the table, denoted by $\mathfrak{M}$. For every $x \in V(T)$, partition (X, Y) of $\beta(x) \setminus D$ and partition $(\widehat{X}, \widehat{Y}_1, \widehat{Y}_2)$ of $\beta(x) \cap D$, we have a table entry $\mathfrak{M}[x, (X, Y), (\widehat{X}, \widehat{Y}_1, \widehat{Y}_2)]$. The order of the computation is done by postorder on T (where the order of computation of entries with the same first argument is arbitrary). Then, the basis corresponds to the case where x is a leaf. In this case, we initialize $\mathfrak{M}[x, (\emptyset, \emptyset), (\emptyset, \emptyset, \emptyset)] = 0$. Now, suppose that x is not a leaf. Then, we use the following recursive formulas:

1. In case x is of type Forget, let y be its child and $v \in \beta(y) \setminus \beta(x)$. Then, we compute $\mathfrak{M}[x, (X, Y), (\widehat{X}, \widehat{Y}_1, \widehat{Y}_2)]$:
   - If $v \in D$, then we take $\min\{\mathfrak{M}[y, (X, Y), (\widehat{X} \cup \{v\}, \widehat{Y}_1, \widehat{Y}_2)], \mathfrak{M}[y, (X, Y), (\widehat{X}, \widehat{Y}_1 \cup \{v\}, \widehat{Y}_2)]\}$.
   - Else, we take $\min\{\mathfrak{M}[y, (X \cup \{v\}, Y), (\widehat{X}, \widehat{Y}_1, \widehat{Y}_2)], \mathfrak{M}[y, (X, Y \cup \{v\}), (\widehat{X}, \widehat{Y}_1, \widehat{Y}_2)]\}$.
2. In case x is of type Introduce, let y be its child and $v \in \beta(x) \setminus \beta(y)$. Then, we compute $\mathfrak{M}[x, (X, Y), (\widehat{X}, \widehat{Y}_1, \widehat{Y}_2)]$:
   - If $v \in X \cup \widehat{X}$, then we take $\mathfrak{M}[y, (X \setminus \{v\}, Y), (\widehat{X} \setminus \{v\}, \widehat{Y}_1 \setminus N_G(v), \widehat{Y}_2 \cup (\widehat{Y}_1 \cap N_G(v)))] + w(v)$.
   - Else, if $v \in Y \cup \widehat{Y}_2$, then we take $\mathfrak{M}[y, (X, Y \setminus \{v\}), (\widehat{X}, \widehat{Y}_1, \widehat{Y}_2 \setminus \{v\})]$.
   - Else, if $v \in N_G(X \cup \widehat{X})$, then we take $\mathfrak{M}[y, (X, Y), (\widehat{X}, \widehat{Y}_1 \setminus \{v\}, \widehat{Y}_2)]$.
   - Else, we take ∞.
3. In case x is of type Join, let y and z be its children. Then, $\mathfrak{M}[x, (X, Y), (\widehat{X}, \widehat{Y}_1, \widehat{Y}_2)]$ equals:

$$\min_{(Y_1^y, Y_1^z) \text{ partition of } \widehat{Y}_1} \{\mathfrak{M}[y, (X, Y), (\widehat{X}, Y_1^y, \widehat{Y}_2 \cup (\widehat{Y}_1 \setminus Y_1^y))] + \mathfrak{M}[z, (X, Y), (\widehat{X}, Y_1^z, \widehat{Y}_2 \cup (\widehat{Y}_1 \setminus Y_1^z))]\}$$

$$-w(X \cup \widehat{X}).$$

Eventually, the algorithm returns the weight stored in $M[\mathsf{root}(T), (\emptyset, \emptyset), (\emptyset, \emptyset, \emptyset)]$, where the matching itself can be retrieved by backtracking its computation (specifically, collecting the vertices inserted into $X \cup \widehat{X}$).

Because for every $x \in V(T)$, $|\beta(x) \cap D| \leq \frac{\mathsf{tw}}{2} + O(1)$, and $|V(T)| \leq O(\mathsf{tw}^{O(1)} \cdot n)$, we derive that the size of $\mathfrak{M}$ is $O(2^{\frac{\mathsf{tw}}{2}} \cdot 3^{\frac{\mathsf{tw}}{2}} \cdot \mathsf{tw}^{O(1)} \cdot n) = O(\sqrt{6}^{\mathsf{tw}} \cdot \mathsf{tw}^{O(1)} \cdot n)$. So, clearly, the computation of all entries corresponding to leaves, Forget nodes and Introduce nodes can be done within this time bound. The computation of all Join nodes can also be done within this time bound by the use of *fast subset convolution* in the exact same manner as it is done for the known exact algorithm for Weighted Dominating Set parameterized by tw (see Section 11.1 in [4]).

Correctness can be proved by straightforward induction on the order of the computation (following the same lines as for the exact algorithm for Weighted Dominating Set parameterized by tw).

▷ Claim 8. Every entry $\mathfrak{M}[x, (X, Y), (\widehat{X}, \widehat{Y}_1, \widehat{Y}_2)]$ stores the minimum weight of a dominating set S of $G[\gamma(x)]$ that satisfies:

- $S \cap \beta(x) = X \cup \widehat{X}$.
- S dominated $\widehat{Y}_1$.

If such a matching does not exist, then the entry stores ∞.

This completes the proof. ◀

For the proof of Theorem 4, we also need the following result.

▶ **Proposition 9** ([17], Corollary). *There exists an $O(n \cdot \mathsf{tw})$-time algorithm that, given a graph G and a tree decomposition $\mathcal{T} = (T, \beta)$ of G of width* tw*, outputs a partition (V_1, V_2) of $V(G)$ such that for every $i \in \{1, 2\}$ and $x \in V(T)$, $|\beta(x) \cap V_i| \leq \frac{\mathsf{tw}}{2} + O(1)$.*

We proceed with the following immediate observation.

▶ **Observation 10.** *Let (G, w) be an instance of* Weighted Dominating Set*, and let $D \subseteq V(G)$. Then,*

1. *Any dominating set $S \subseteq V(G)$ of G dominates both D and $V(G) \setminus D$*
2. *Let $S_1 \subseteq V(G)$ dominate D, and $S_2 \subseteq V(G)$ dominate $V(G) \setminus D$. Then, $S_1 \cup S_2$ is a dominating set of G.*

Now, we are ready to conclude the correctness of Theorem 4.

Proof of Theorem 4. We first describe the algorithm. Let $(G, w, \mathcal{T})$ be an instance of Weighted Dominating Set parameterized by tw. Due to Proposition 3, we can suppose that $\mathcal{T}$ is nice. First, we call the algorithm of Proposition 9 with $(G, w, \mathcal{T})$ as input, and let (V_1, V_2) denote its outputs. Then, we call the algorithm of Lemma 7 twice, once with $(G, w, \mathcal{T}, V_1)$ as input and once with $(G, w, \mathcal{T}, V_2)$ as input, and let S_1 and S_2 denote their outputs. We return $S = S_1 \cup S_2$.

Clearly, due to Proposition 9 and Lemma 7, the algorithm runs in $O(\sqrt{6}^{\mathsf{tw}} \cdot \mathsf{tw}^{O(1)} \cdot n)$ time. For correctness, first note that due to the second item in Observation 10, the output set S is a dominating set of G. Moreover, due to the first item in Observation 10, the optimums of $(G, w, \mathcal{T}, V_1)$ and $(G, w, \mathcal{T}, V_1)$ as instances of Weighted Half-Width Domination are both bounded from above by the optimum of $(G, w, \mathcal{T})$ as an instance of Weighted Dominating Set. Hence, both of $w(S_1), w(S_2)$ are bounded from above by the optimum of $(G, w, \mathcal{T})$ as an instance of Weighted Dominating Set, which implies that $w(S)$ is bounded from above by twice the optimum of $(G, w, \mathcal{T})$ as an instance of Weighted Dominating Set. This completes the proof. ◀

4 Parameter: Size of Vertex Modulator to Constant Treewidth

Similarly to the proof in Section 3, will use a translation of an instance of (Weighted) Dominating Set into two instances of two easier problems, to attain a 2-approximation algorithm for (Weighted) Dominating Set. Here, however, the translation is somewhat different.

▶ **Theorem 11.** *For any fixed constant $d \geq 1$, the* Weighted Dominating Set *problem parameterized by tw_d admits a 2-approximation $O(2^{\mathsf{tw}_d} \cdot n)$-time algorithm.*

Towards the proof of this theorem, we first define the two easier problems that we will aim to solve.

▶ **Definition 12** (Modulator Domination). *Let $d \in \mathbb{N} \cup \{0\}$. In the* Weighted d-Modulator Domination *problem, the input consists of a graph G, a vertex-weight function $w : V(G) \to \mathbb{N}$, and a subset $M \subseteq V(G)$ such that the treewidth of $G - M$ is at most d. The objective is to compute a subset $S \subseteq V(G)$ of minimum weight that dominates M.*

▶ **Definition 13** (Decomposition Domination). *Let $d \in \mathbb{N} \cup \{0\}$. In the* Weighted d-Decomposition Domination *problem, the input consists of a graph G, a vertex-weight function $w : V(G) \to \mathbb{N}$, and a subset $M \subseteq V(G)$ such that the treewidth of $G - M$ is at most d. The objective is to compute a subset $S \subseteq V(G)$ of minimum weight that dominates $V(G) \setminus M$.*

To reuse the result for Modulator Domination in Section 5, we require a slight generalization of the problem, defined as follows.

▶ **Definition 14** (Generalized Modulator Domination). *Let $d \in \mathbb{N} \cup \{0\}$. In the* Generalized Weighted d-Modulator Domination *problem, the input consists of a graph G, a vertex-weight function $w : V(G) \to \mathbb{N}$, and a subset $M \subseteq V(G)$ such that the treewidth of $G - M$ is at most d. The objective is to compute, for every subset $A \subseteq M$, a subset $S_A \subseteq V(G)$ of minimum weight that dominates A.*

Next, we present our algorithms for Generalized Weighted d-Modulator Domination and Weighted d-Decomposition Domination. For the Generalized Weighted d-Modulator Domination problem, we will use the following result.

▶ **Proposition 15** ([4], Implicit). *The* Generalized Weighted Set Cover *problem admits an $O(2^n \cdot m)$-time algorithm, where n is the size of the universe and m is the size of the set-family.*

▶ **Lemma 16.** *The* Generalized Weighted d-Modulator Domination *problem admits an $O(2^{|M|} \cdot n)$-time algorithm.*

Proof. We first describe the algorithm. Let (G, w, M) be an instance of the Generalized Weighted d-Modulator Domination problem. Then, we construct an instance $(U, \mathcal{F}, w')$ of Generalized Weighted Set Cover as follows:

- $U = M$.
- $\mathcal{F} = \{N[v] \cap M : v \in V(G)\}$.
- For every $F \in \mathcal{F}$, let v_F be a vertex of minimum weight among the vertices $v \in V(G)$ that satisfy $F = N[v] \cap M$, and define $w'(F) = w(v_F)$.

We call the algorithm of Proposition 15 with $(U, \mathcal{F}, w')$ as input, and, for every $A \subseteq U$, let $\mathcal{S}_A$ be its output. Then, for every $A \subseteq M$, we return $S_A = \{v_F : F \in \mathcal{S}_A\}$.

Clearly, due to Proposition 15, the algorithm runs in $O(2^{|M|} \cdot n)$ time. For correctness, consider some $A \subseteq M$. Observe that, on the one hand, if $B \subseteq V(G)$ dominates A, then $\mathcal{B} = \{N[v] \cap A : v \in B\} \subseteq \mathcal{F}$ covers A and $w(B) \geq w'(\mathcal{B})$. On the other hand, if $\mathcal{B} \subseteq \mathcal{F}$ covers A, then $B = \{v_F : F \in \mathcal{A}\} \subseteq V(G)$ dominates A, and $w'(\mathcal{B}) = w(B)$. This completes the proof. ◀

For the Weighted d-Decomposition Domination problem, we will use Proposition 6 and the following result.

▶ **Proposition 17** ([2]). *There exists an algorithm that, given a graph G, outputs a tree decomposition of G of width $t = \mathsf{tw}(G)$ in $t^{O(t^3)} \cdot n$ time.*

▶ **Lemma 18.** *The* Weighted d-Decomposition Domination *problem admits an $O(2^{|M|} \cdot n)$-time algorithm.*

Proof. We first describe the algorithm. Let (G, w, M) be an instance of the Weighted d-Decomposition Domination problem. Then, for every subset $L \subseteq M$, we construct an instance $I_L = (G_L, w_L, \mathcal{T}_L)$ of Weighted Dominating Set parameterized by tw as follows:

- Let $V(G_L) = (V(G) \setminus M) \cup \{x\}$ for $x \notin V(G)$, and $E(G_L) = E(G - M) \cup \{\{x, v\} : v \in N_G(L) \setminus M\}$. That is, we construct G_L from G by removing the vertices in M and the edges incident to them, and adding a new vertex x adjacent to all of the vertices in $N_G(L) \setminus M$.
- For every $v \in V(G_L)$, define $w_L(v) = w(v)$ if $v \in V(G) \setminus M$, and $w_L(v) = w(L)$ otherwise (for $v = x$).
- Use the algorithm of Proposition 17 with G_L as input, and let $\mathcal{T}_L$ be its output.

Let $\mathcal{I} = \{I_L : L \subseteq M\}$. For every $I_L \in \mathcal{I}$, we call the algorithm of Proposition 6 with I_L as input, let S'_L be its output, and define S_L as S'_L if $x \notin S'_L$ and $S'_L \cup L$ otherwise. Let $\mathcal{S} = \{S_L : L \subseteq M\}$. Then, we return the set S of minimum-weight with respect to w among the sets in $\mathcal{S}$.

For the time complexity analysis, observe that $|\mathcal{I}| = 2^{|M|}$. Moreover, observe that for every $L \subseteq M$, $\mathsf{tw}(G_L) \leq \mathsf{tw}(G - M) + 1 \leq d + 1$; hence, each call to the algorithm of Proposition 17 runs in $(d+1)^{O((d+1)^{d+1})} \cdot |V(G_L)| \leq O(n)$ time, and each call to the algorithm of Proposition 6 runs in $O(3^{d+1} \cdot |V(G_L)|) \leq O(n)$ time. Thus, the total running time of our algorithm is $O(2^{|M|} \cdot n)$.

Now, we turn to consider the correctness of the algorithm. To this end, consider some subset $L \subseteq M$. On the one hand, consider some subset $A \subseteq V(G)$ that satisfies $A \cap M = L$ and A dominates $V(G) \setminus M$. Let $A' = (A \setminus M) \cup \{x\}$. Then, $A \setminus M$ dominates $V(G_L) \setminus N_G(L)$ and x dominates $N_G(L)$, hence A' dominates $V(G_L)$, and our definition of w_L directly implies that $w(A) = w_L(A')$. On the other hand, consider some subset $A' \subset V(G_L)$ that dominates $V(G_L)$. Then, define A as A' if $x \notin A'$ and $A' \cup L$ otherwise. So, it is easy to see that A dominates $V(G) \setminus M$ and $w_L(A') = w(A)$.

We conclude that, on the one hand, if $A \subseteq V(G)$ dominates M, then, for $L = A \cap M$, a minimum-weight dominating set of G_L with respect to w_L is of weight $w(A)$. So, the output dominating set cannot have weight larger than $w(A)$. On the other hand, for every $L \subseteq M$, the minimum weight of a dominating set of G_L with respect to w_L is bounded from below by the minimum weight of a dominating set of G with respect to w. So, obviously, the output dominating set cannot have weight larger than the minimum one. This completes the proof. ◀

Now, we are ready to conclude the correctness of Theorem 11.

Proof of Theorem 11. We first describe the algorithm. Let (G, w, M) be an instance of Weighted Dominating Set parameterized by tw_d. Then, we call the algorithms of Lemmas 16 and 18 with (G, w, M) as input, and let S_1 and S_2 denote their outputs. We return $S = S_1 \cup S_2$.

Clearly, due to Lemmas 16 and 18, and since $|M| = \mathsf{tw}_d$, the algorithm runs in $O(2^{\mathsf{tw}_d} \cdot n)$ time. For correctness, first note that due to the second item in Observation 10, the output set S is a dominating set of G. Moreover, due to the first item in Observation 10, the optimum of (G, w, M) as an instance of Weighted d-Modulator Domination (or Weighted d-Decomposition Domination) is bounded from above by the optimum of (G, w, M) as an instance of Weighted Dominating Set. Hence, both of $w(S_1), w(S_2)$ are bounded from above by the optimum of (G, w, M) as an instance of Weighted Dominating Set, which implies that $w(S)$ is bounded from above by twice the optimum of (G, w, M) as an instance of Weighted Dominating Set. This completes the proof. ◀

5 Parameter: Vertex Cover Number

In this section, we prove that in the case of $\mathsf{vc} = \mathsf{tw}_0$, we can attain an exact algorithm with the same running time as in Theorem 11.

▶ **Theorem 19.** *The* Weighted Dominating Set *problem parameterized by* vc *admits a* $O(2^{\mathsf{vc}} \cdot n)$*-time algorithm.*

Proof. We suppose that the input also consists of a subset $M \subseteq V(G)$ that is a vertex cover of G of size vc, since such a subset can be easily computed in $O(2^{vc} \cdot n)$ time [8, 4]. Now, we describe the algorithm. Let (G, w, M) be an instance of Weighted Dominating Set parametrized by vc. We perform the following steps:

1. Call the algorithm of Lemma 16 with (G, w, M) as input of Weighted 0-Modulator Domination. Let $\{\widetilde{S}_A : A \subseteq M\}$ be its output.
2. For every $A \subseteq M$:
   a. Let $\widehat{S}_A = A \cup (V(G) \setminus (N_G(A) \cup M))$. That is, $\widehat{S}_A$ is the union of A and the set of vertices in the independent set $V(G) \setminus M$ that are not dominated by the vertices in A.
   b. Let $S_A = \widehat{S}_A \cup \widetilde{S}_{M \setminus N_G[\widehat{S}_A]}$. Notice that $M \setminus N_G[\widehat{S}_A]$ is the set of vertices in M that are not dominated by the vertices in $\widehat{S}_A$.
3. Return the set S of minimum weight among the sets in $\{S_A : A \subseteq M\}$.

Clearly, due to Lemma 16, the algorithm runs in $O(2^{\mathsf{vc}} \cdot n)$ time. Moreover, it is clear that the output set S is a dominating set of G. So, it remains to show that S is of minimum weight among all dominating sets of G. To this end, let $S^\star$ be a dominating set of G of minimum weight. Consider the iteration of the algorithm that corresponds to $A^\star = S^\star \cap U$. Notice that, since $S^\star$ dominates $V(G) \setminus M$ which is an independent set, it must hold that $V(G) \setminus (N_G(A^\star) \cup M) \subseteq S^\star$. So, $\widehat{S}_{A^\star} \subseteq S^\star$. Further, since $S^\star$ dominates $M \setminus N_G[\widehat{S}_{A^\star}]$, we have that $S^\star \setminus \widehat{S}_{A^\star}$ dominates $M \setminus N_G[\widehat{S}_{A^\star}]$. By the correctness of the algorithm of Lemma 16, this implies that $w(\widetilde{S}_{M \setminus N_G[\widehat{S}_{A^\star}]}) \leq w(S^\star \setminus \widehat{S}_{A^\star})$. Thus, we conclude that $w(S) \leq w(S_{A^\star}) \leq w(S^\star)$. ◀

Additionally, we observe that the time complexity in Theorem 11 is tight. Due to lack of space, the proof is deferred to the full version of this paper.

▶ **Observation 20.** *Under any of the SETH and the Set Cover Conjecture, the* Dominating Set *problem parameterized by* vc *cannot be solved in* $O((2-\epsilon)^{\mathsf{vc}} \cdot n)$ *time for any fixed* $\epsilon > 0$.

6 Parameter: Feedback Edge Set Number: FPT Algorithm

In this section, we first prove a combinatorial result (stated in Lemma 22). In particular, this result implies a parameterized algorithm where the basis of the exponent is smaller than 3 (stated in Theorem 29). For our combinatorial result, we will use the following proposition.

▶ **Proposition 21** ([3])**.** *The treewidth of a cactus is at most* 2.

▶ **Lemma 22.** *For any graph* G, $\mathsf{tw}_2(G) \leq \frac{\mathsf{fes}(G)}{2}$. *Moreover, there exists an algorithm that, given a graph* G, *outputs a subset* $M \subseteq V(G)$ *such that* $|M| \leq \frac{\mathsf{fes}(G)}{2}$ *and* $\mathsf{tw}(G - M) \leq 2$ *in* $O(\mathsf{fes}(G) + n)$ *time.*

The idea behind the algorithm presented in the proof is quite simple (though, perhaps, if we did not demand it to run in $O(\mathsf{fes}(G) + n)$ time, it could have been further simplified). Specifically, we scan a depth-first search (DFS) tree T of G from top to bottom. For each vertex that we remove (and insert into M), we aim to argue that at least two edges in $F = E(G) \setminus E(T)$ have become "irrelevant" – that is, not part of any cycle. To identify which vertices to remove, we maintain a variable e, which stores an edge from F whose "top" is above (or equal to) and whose "bottom" is below (or equal to) the vertex currently under consideration, and, most importantly, which is still "relevant". When no such edge exists, it stores nil. In particular, we notice two situations where we can (and it suffices) to remove a vertex: first, when it is the top of two edges from F, and second, when it is the top of an edge from F and e is some other edge from F. We now proceed to present the formal description of the algorithm and its proof.

Proof of Lemma 22. To describe the algorithm, let G be a graph. Without loss of generality, we suppose that G is connected, else we can consider each of its connected components separately. We compute a DFS tree T of G. Let $F = E(G) \setminus E(T)$, and note that $|F| = \mathsf{fes}(G)$. Given an edge $e \in F$, we refer to the *top* and *bottom* of e as the endpoint of e that is closer to the root of T and the other endpoint of e, respectively. (Since T is a depth-first search tree, the terms top and bottom are uniquely defined.)

Initialize $M = \emptyset$ and $e = \mathsf{nil}$. For every $v \in V(T)$ where T is traversed in preorder, we perform the following computation:

1. If v is the bottom of e (in this case, $e \neq \mathsf{nil}$), update $e = \mathsf{nil}$.
2. If v is not the top of any edge in F, we proceed to the next iteration.
3. If either v is the top of at least two edges in F or $e \neq \mathsf{nil}$, then:
   a. Insert v into M.
   b. Update $e = \mathsf{nil}$.
   c. Proceed to the next iteration.
4. Update e to be the edge in F whose top is v, and prioritize the preorder traversal to first visit the vertices on the subpath of T from v to the bottom of e. (In case a previous prioritization exists, override it.)

At the end, we return the set M.

Clearly, the algorithm runs in $O(n + m) = O(\mathsf{fes}(G) + n)$ time.

We now turn to consider the correctness of the algorithm. Towards that, we define the following terminology. Given an edge $e \in F$, let the *span* of e be the subpath of T from the top to bottom of e, and let the *truncated span of* e be the subpath that results from the removal of the bottom of e from the span of e. We say that two distinct edges $e, e' \in F$ have a *conflict* if the top of one of them belongs to truncated span of the other. Given an edge $e \in F$ and a subset $M \subseteq V(G)$, we say that e is *active* in M if $G - M$ contains a cycle that traverses e. Observe that all of the edges in F are active in $\emptyset$.

Towards the proof of our main inductive claim, we present the following claim.

▷ Claim 23. Let C be a cycle in G, $e \in E(C) \cap F$, and suppose that C is not the cycle formed by e and its span. Then, C contains an edge $e' \in F \setminus \{e\}$ that has a conflict with e, and whose top is either the top of e or an ancestor of it.

Proof. Let t and b be the top and bottom of e, respectively. Targeting a contradiction, we assume that C does not contain an edge $e' \in F \setminus \{e\}$ that has a conflict with e, and whose top is either t or an ancestor of t. Let P denote the subpath of C between t and b that does not contain e. Due to our assumption, this path cannot contain an edge between t or an ancestor of t and a descendant of t, with the exception of the edge between t and its children in T, because such an edge must belong to F and have a conflict with e. Due to this, and because T is a depth-first tree, P cannot contain an edge between a vertex that is not a descendant of t and a descendant of t, with the exception of the edge between t and its child that belongs to the span of e, which we denote by c. So far, we conclude that P does not contain any ancestor of t and that it contains the edge $\{t, c\}$. However, again, because T is a depth-first tree, P also cannot contain an edge between a vertex that is a descendant of a vertex, say, x, that belongs to the span of e and a vertex that is neither x nor an ancestor of x. In turn, this implies that P is equal to the span of e, which is a contradiction to the supposition of the claim that C is not the cycle formed by e and its span. ◁

Now, we are ready to present our main inductive argument.

▷ Claim 24. Consider an iteration of the preorder traversal. Let M' be the set M at the end of this iteration. Let e' denote the value of e at the end of this iteration. Let v be the vertex traversed in this iteration. Then:

1. The set M' does not contain any descendant of v.
2. There do not exist two edges in F that are active in M', have a conflict and the top of each one of them is either v or an ancestor of v in T.
3. If $e' = \mathsf{nil}$, then there does not exist an edge in F that is active in M' and such that v belongs to the truncated span of that edge.
4. If $e' \neq \mathsf{nil}$, then: (i) v belongs to the truncated span of e'; (ii) there does not exist an edge in F other than e' that is active in M' and such that v belongs to the truncated span of that edge; (iii) M' does not contain any vertex from the span of e'. (In particular due to item 1 and (iii), e' is active, and this is witnessed by the cycle formed by e' and its span.)

Proof. We use induction on the preorder traversal. Consider the first iteration, where v is the root of T and, hence, the only edges in F such that v belongs to their span are those that have v as their top. Then, all of the items in the claim directly follow from the pseudocode.

Now, consider an iteration that is not the first, and suppose that the claim is correct up to this iteration. By the inductive hypothesis (item 1) and the pseudocode, it should be clear that item 1 of the claim holds. Let M'' and e'' denote the values of M and e at the beginning of the iteration. Let u be the parent of v in T. By the inductive hypothesis (item 2), there do not exist two edges in F that are active in M'', have a conflict and the top of each one of them is an ancestor of u in T. Yet, to prove item 2 of the claim, we still need to argue that there do not exist two edges in F that are active in M', have a conflict, the top of one of them is v, and the top of the other is either v or an ancestor of v. Note that if there exist two such edges, then v belongs to the truncated span of both of these edges. We consider the two following cases.

- First, suppose that $e'' = \mathsf{nil}$. Then, by the inductive hypothesis (item 3), there does not exist an edge in F that is active in M'' and such that u belongs to the truncated span of that edge. So, the only edges that are active in M'' and such that v belongs to their span are those that have v as their top. If v is not the top of any edge in F, then $e' = \mathsf{nil}, M' = M''$, and items 2 and 3 of the claim follow. If v is the top of at least two edges in F, then $e' = \mathsf{nil}, M' = M'' \cup \{v\}$ (so, these edges are non-active in M'), and items 2 and 3 of the claim follow. If v is the top of exactly one edge in F, then this edge is e' (and $M' = M''$), and, hence, items 2 and 4 of the claim follow.
- Second, suppose that $e'' \neq \mathsf{nil}$. Then, by the inductive hypothesis (item 4), u belongs to the truncated span of e'', there does not exist an edge in F other than e'' that is active in M'' and such that u belongs to the truncated span of that edge, and M'' does not contain any vertex from the span of e''. We further consider the three following sub-cases:
  1. First, suppose that v is the bottom of e''. This implies that the only edges that are active in M'' and such that v belongs to their span are those that have v as their top. Then, e is updated to be nil in the first step of the iteration, and the proof proceeds as in the first case.
  2. Second, suppose that v is neither the bottom of e'' nor the top of any edge in F. This implies that v belongs to the truncated span of e'', and that there does not exist an edge in F other than e'' that is active in M'' and such that v belongs to the span of that edge. As $e' = e''$ and $M' = M''$, items 2 and 4 of the claim follow.
  3. Third, suppose that v is not the bottom of e'', and that v is the top of at least one edge in F. Then, $e' = \mathsf{nil}$ and $M' = M'' \cup \{v\}$. Hence, there does not exist an edge in F that is active in M' and has v as its top. Hence, item 2 of the claim follows, and to complete the proof of item 4 of the claim, it suffices to show that e'' is non-active in M'.

Targeting a contradiction, suppose that e'' is active in M', and let t and b denote its top and bottom, respectively. Then, there exists a cycle C in $G - M'$ that contains e''. In particular, there exists a path P in $G - M'$ between t and b that does not contain e''. Due to Claim 23, if P is not equal to the span of e'', then C contains an edge $\widehat{e} \in F \setminus \{e''\}$ that has a conflict with e'' and whose top is either t or an ancestor of t, and because this edge belongs to C (which exists in $G - M'$), it must be active in M'; however, this is a contradiction to item 2 of the claim. Thus, P is equal to the span of e'', which is a contradiction, since v belongs to this span as well as to M'. So, e'' is non-active in M'.

This completes the proof. ◁

We proceed to prove the following claim, which will imply the desired bound the size of M.

▷ Claim 25. Consider an iteration of the preorder traversal. Let M' be the set M at the end of this iteration. Suppose that in this iteration, the vertex v was inserted into M'. Then, there exist two distinct edges in F that are active in $M' \setminus \{v\}$ but are non-active in M'.

Proof. Let e'' denote the value of e at the start of this iteration. Then, one of the two following cases holds.

Case I. Suppose that v is the top of at least two edges in F, say, e^v_1 and e^v_2. Then, due to item 1 of Claim 24, both of these edges are active in $M' \setminus \{v\}$ (witnesses by the cycles formed by these edges and their spans). However, both of these edges clearly become non-active in M'.

Case II. Suppose that $e'' \neq \mathsf{nil}$ and v is the top of exactly one edge in F, denoted by e^v. Note that $e'' \neq e^v$, since the value of e is updated when its top is traversed (and v is only being traversed in the current iteration, after e already holds e''). As in Case I, e^v is active in $M' \setminus \{v\}$ but becomes non-active in M'. By item 4 of Claim 24 with respect to the previous iteration, e'' is active in $M' \setminus \{v\}$, and by the same item with respect to the current iteration, e'' is non-active in M'.

In either case, we conclude that the claim holds. ◁

In particular, from Claim 25 we conclude that $|M| \leq \frac{|F|}{2} = \frac{\mathsf{fes}(G)}{2}$. (For every vertex inserted into M, at least two edges in F that are active at that moment become non-active, and they never become active again later).

In order to bound the treewidth of $G - M$, we turn to prove several additional claims.

▷ Claim 26. Let $X \subseteq V(G)$. Then, $\{e \in F : e \text{ is active in } X\}$ is a feedback edge set of $G - X$.

Proof. The claims directly follows from the definition of active edges, and because F is a feedback edge set of G. ◁

▷ Claim 27. Let $X \subseteq V(G)$. Let C, C' be two distinct cycles in $G - X$ that have at least two vertices in common. Then, there exist two distinct edges $e, e' \in F$ that are active in X and have a conflict.

Proof. By Claim 23 and since C, C' belong to $G - X$, we can assume that C and C' are the cycles that consist of some edges $e, e' \in F$ and their spans, respectively, else the proof is complete. Since C and C' have at least two vertices in common, the intersection of the spans of e and e' must be of size at least 2. However, this implies that the top of one of them must belong to the truncated span of the other, and hence they have a conflict. ◁

▷ Claim 28. There do not exist two distinct edges $e, e' \in F$ that are active in M and have a conflict.

Proof. The claim directly follows from item 2 of Claim 24 by considering the iterations in which the leaves of T were traversed. ◁

From Claims 27 and 28, we derive that $G - M$ is a cactus graph. So, by Proposition 21, we conclude that its treewidth is at most 2. This completes the proof. ◀

▶ **Theorem 29.** *The* Weighted Dominating Set *problem parameterized by* fes *admits an* $O(3^{\frac{\mathsf{fes}}{2}} \cdot n)$*-time algorithm.*

Proof. To describe the algorithm, let (G, w) be an instance of Weighted Dominating Set. Then, we call the algorithm of Lemma 22, and let M be its output. So, $|M| \leq \frac{\mathsf{fes}(G)}{2}$ and $\mathsf{tw}(G - M) \leq 2$. Afterwards, we call the algorithm of Proposition 17 with $G - M$ as input, and let $\mathcal{T}'$ be its output. So, $\mathcal{T}'$ is a tree decomposition of width at most 2 of $G - M$. We insert M into each of the bags of $\mathcal{T}'$ to attain a tree decomposition $\mathcal{T}$ of G of width at most $|M| + 2 \leq \frac{\mathsf{fes}(G)}{2} + 2$. Lastly, we call the algorithm of Proposition 6 with $(G, w, \mathcal{T})$ as input, and return its result.

Clearly, correctness is immediate. As for the time complexity, observe that the calls to the algorithms of Lemma 22 and Propositions 17 and 6 run in $O(\mathsf{fes}(G) + n)$, $2^{O(2^3)} \cdot n = O(n)$ and $O(3^{\frac{\mathsf{fes}(G)}{2}+2} \cdot n) \leq O(3^{\frac{\mathsf{fes}(G)}{2}} \cdot n)$ times, respectively. Thus, the total running time of our algorithm is $O(3^{\frac{\mathsf{fes}(G)}{2}} \cdot n)$. ◀

7 Conclusion and Future Directions

We presented algorithmic results that sidestep time complexity barriers for Dominating Set. For this purpose, we incorporated approximation for the parameters solution size and treewidth, larger parameterization for the parameters vertex cover and feedback edge set compared to treewidth, or both for the parameter vertex modulator to constant treewidth compared to treewidth.

Extension of Our Approaches. While we have focused on Dominating Set, we believe that some of our approaches might be applicable to other problems as well. For example, consider the Graph Coloring problem, where, given a graph G and an integer $q \geq 3$, the objective is to determine whether G admits a proper coloring in q colors. Under the SETH, Graph Coloring cannot be solved in $(q - \epsilon)^{\mathsf{tw}} \cdot n^{O(1)}$ time for any fixed $\epsilon > 0$ [16]. Then, we follow a simplification of the approach we presented in Section 4. Briefly, the idea is to consider two problems: one problem concerns the graph induced by the modulator, and the other problem concerns the rest of the graph. So, suppose we are given a subset $M \subseteq V(G)$ such that the treewidth of $G - M$ is at most d (where d is a fixed constant). On the one hand, we solve Graph Coloring on $G[M]$ in $2^{|M|} \cdot |M|^{O(1)}$ time using the algorithm in [15], and on the other hand, we solve Graph Coloring on $G - M$ in $n^{O(1)}$ time based on straightforward dynamic programming. We consider the color sets used by the two solutions to be disjoint, thereby obtaining a 2-approximate solution in $2^{\mathsf{tw}_d} \cdot n^{O(1)}$ time. Essentially the same approach works for INDEPENDENT SET as well, where, given a graph G and a non-negative integer k, the objective is to determine whether G admits an independent set of size at least k. Under the SETH, INDEPENDENT SET cannot be solved in $(2-\epsilon)^{\mathsf{tw}} \cdot n^{O(1)}$ time for any fixed $\epsilon > 0$ [16]. On the one hand, we solve INDEPENDENT SET on $G[M]$ in $1.19997^{|M|} \cdot |M|^{O(1)}$ time using the algorithm in [24], and on the other hand, we solve INDEPENDENT SET on $G - M$ in $n^{O(1)}$ time based on straightforward dynamic programming. We output the largest among the two solutions, thereby obtaining a 2-approximate solution in $1.19997^{\mathsf{tw}_d} \cdot n^{O(1)}$ time.

Directions for Future Research. Firstly, we find the questions of improvements of the performance of our algorithms (in terms of running times and approximation ratios) interesting. In particular, does DOMINATING SET admit a 2-approximation $O(2^{\mathsf{tw}} \cdot n)$-time algorithm, or a $(1+\epsilon)$-approximation $O(2^{\mathsf{tw}} \cdot n)$-time algorithm for any fixed $\epsilon > 0$? Additionally, we have the following questions regarding DOMINATING SET:

1. Prove or refute the following conjecture:

▶ **Conjecture 30.** *Under the SETH, there exists a fixed constant $d \in \mathbb{N}$ such that* (WEIGHTED) DOMINATING SET *cannot be solved in $(3-\epsilon)^{\mathsf{tw}_d} \cdot n^{f(d)}$ time for any fixed constant $\epsilon > 0$ and function f of d, where tw_d is the minimum size of a vertex set whose deletion from G results in a graph of treewidth at most d.*

2. Study DOMINATING SET parameterized by the solution size plus the distance (e.g., number of vertex or edge deletions or contractions) to graph classes where it belongs to FPT, particularly planar graphs and claw-free graphs.
3. Conduct a similar study for problems beyond DOMINATING SET. Here, possibly and as argued above, the ideas presented in this article can be re-used.

References

1 Saeed Akhoondian Amiri, Patrice Ossona de Mendez, Roman Rabinovich, and Sebastian Siebertz. Distributed domination on graph classes of bounded expansion. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, SPAA '18, pages 143–151, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3210377.3210383.

2 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

3 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.

4 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

5 Marek Cygan, Lukasz Kowalik, and Mateusz Wykurz. Exponential-time approximation of weighted set cover. *Inf. Process. Lett.*, 109(16):957–961, 2009.

6 Marek Cygan, Geevarghese Philip, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Dominating set is fixed parameter tractable in claw-free graphs. *Theoretical Computer Science*, 412(50):6982–7000, November 2011. doi:10.1016/j.tcs.2011.09.010.

7 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633, 2014.

8 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer Publishing Company, Incorporated, 2013.

9 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004. `doi:10.1016/j.tcs.2004.05.009`.

10 Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, July 1998. `doi:10.1145/285055.285059`.

11 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Linear kernels for (connected) dominating set on h-minor-free graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 82–93, USA, 2012. Society for Industrial and Applied Mathematics.

12 Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM Journal on Computing*, 36(2):281–309, 2006. `doi:10.1137/S0097539702419649`.

13 Danny Hermelin, Matthias Mnich, Erik Jan van Leeuwen, and Gerhard J. Woeginger. Domination when the stars are out. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 462–473, 2011. `doi:10.1007/978-3-642-22006-7_39`.

14 Tohru Kikuno, Noriyoshi Yoshida, and Yoshiaki Kakuda. A linear algorithm for the domination number of a series-parallel graph. *Discret. Appl. Math.*, 5:299–311, 1983.

15 Mikko Koivisto. An o*(2ˆn) algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 583–590, 2006.

16 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2), April 2018. `doi:10.1145/3170442`.

17 Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Fpt-approximation for FPT problems. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 199–218, 2021. `doi:10.1137/1.9781611976465.14`.

18 Dániel Marx. Parameterized Complexity of Independence and Domination on Geometric Graphs. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation*, pages 154–165, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

19 Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1065–1075, USA, 2010. Society for Industrial and Applied Mathematics.

20 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5), August 2019. `doi:10.1145/3325116`.

21 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic Programming on Tree Decompositions Using Generalised Fast Subset Convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009*, pages 566–577, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

22 Johan M.M. van Rooij and Hans L. Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164, 2011. `doi:10.1016/j.dam.2011.07.001`.

23 Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001. URL: `http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-65367-7`.

24 Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Inf. Comput.*, 255:126–146, 2017.