Approximate Turing Kernelization for Problems Parameterized by Treewidth

Eva-Maria C. Hols [ID]
Department of Computer Science, Humboldt-Universität zu Berlin, Germany
eva-maria.hols@fkie.fraunhofer.de

Stefan Kratsch [ID]
Department of Computer Science, Humboldt-Universität zu Berlin, Germany
kratsch@informatik.hu-berlin.de

Astrid Pieterse [ID]
Department of Computer Science, Humboldt-Universität zu Berlin, Germany
astrid.pieterse@informatik.hu-berlin.de

Abstract

We extend the notion of lossy kernelization, introduced by Lokshtanov et al. [STOC 2017], to approximate Turing kernelization. An α-approximate Turing kernel for a parameterized optimization problem is a polynomial-time algorithm that, when given access to an oracle that outputs c-approximate solutions in $\mathcal{O}(1)$ time, obtains an $\alpha \cdot c$-approximate solution to the considered problem, using calls to the oracle of size at most $f(k)$ for some function f that only depends on the parameter.

Using this definition, we show that INDEPENDENT SET parameterized by treewidth ℓ has a $(1+\varepsilon)$-approximate Turing kernel with $\mathcal{O}(\frac{\ell^2}{\varepsilon})$ vertices, answering an open question posed by Lokshtanov et al. [STOC 2017]. Furthermore, we give $(1+\varepsilon)$-approximate Turing kernels for the following graph problems parameterized by treewidth: VERTEX COVER, EDGE CLIQUE COVER, EDGE-DISJOINT TRIANGLE PACKING and CONNECTED VERTEX COVER.

We generalize the result for INDEPENDENT SET and VERTEX COVER, by showing that all graph problems that we will call *friendly* admit $(1+\varepsilon)$-approximate Turing kernels of polynomial size when parameterized by treewidth. We use this to obtain approximate Turing kernels for VERTEX-DISJOINT H-PACKING for connected graphs H, CLIQUE COVER, FEEDBACK VERTEX SET and EDGE DOMINATING SET.

1 Introduction

Many important computational problems are NP-hard and, thus, they do not have efficient algorithms unless P = NP. At the same time, it is well known that *efficient preprocessing* can greatly speed up (exponential-time) algorithms for solving NP-hard problems. The notion of a *kernelization* from parameterized complexity has allowed a rigorous and systematic study of this important paradigm. The central idea is to relate the effectiveness of preprocessing to the structure of the input instances, as quantified by suitable parameters.

A *parameterized problem* consists of any (classical) problem together with a choice of one or more parameters; we use (x, k) to denote an instance with input data x and parameter k. A *kernelization* is an efficient algorithm that on input of (x, k) returns an equivalent instance

(x', k') of size upper bounded by $f(k)$, where f is a computable function. For a polynomial kernelization we require that the size bound $f(k)$ is polynomially bounded in k. The study of which parameterized problems admit (polynomial) kernelizations has turned into a very active research area within parameterized complexity (see, e.g., [1, 5, 7, 8, 16, 23, 27, 28, 29, 31, 35] and the recent book [17]). An important catalyst for this development lies in the ability to prove lower bounds for kernelizations, e.g., to conditionally rule out polynomial kernels for a problem, which was initiated through work of Bodlaender et al. [4] and Fortnow and Santhanam [18].

Unfortunately, the lower bound tools have also revealed that many fundamental parameterized problems do not admit polynomial kernelizations (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ and the polynomial hierarchy collapses). These include a variety of problems like CONNECTED VERTEX COVER [12], DISJOINT CYCLE PACKING [6], MULTICUT [11], and k-PATH [4] parameterized by solution size, but also essentially any NP-hard problem parameterized by width parameters such as treewidth. This has motivated the study of relaxed forms of kernelization, notably Turing kernelization [3] and lossy (or approximate) kernelization [30].

Given an input (x, k), a *Turing kernelization* may create $|x|^{\mathcal{O}(1)}$ many instances of size at most $f(k)$ each, and the answer for (x, k) may depend on solutions for all those instances. This is best formalized as an efficient algorithm that solves (x, k) while being allowed to ask questions of size at most $f(k)$ to an oracle. A priori, this is much more powerful than regular kernelization, which creates only a single output instance. Nevertheless, there are only few polynomial Turing kernelizations known for problems without (regular) polynomial kernelization (e.g., [3, 26, 25, 34]). Moreover, a hardness-based approach of Hermelin et al. [22] gives evidence that many problems are unlikely to admit polynomial Turing kernels.

More recently, Lokshtanov et al. [30] proposed a framework dedicated to the study of *lossy kernelization*. This relaxes the task of the kernelization by no longer requiring that an optimal solution to the output (x', k') yields an optimal solution for (x, k). Instead, for an α-approximate kernelization any c-approximate solution to (x', k') can be lifted to an $\alpha \cdot c$-approximate solution for (x, k). Amongst others, they show that CONNECTED VERTEX COVER and DISJOINT CYCLE PACKING admit approximate kernelizations. In contrast, they were able to show, e.g., that k-PATH has no α-approximate kernelization for any $\alpha \geq 1$ (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$). Subsequent works have shown approximate kernelizations for other problems [13, 14, 32], in particular, further problems with connectivity constraints, which are often an obstruction for the existence of polynomial kernelizations.

Lokshtanov et al. [30] ask whether INDEPENDENT SET parameterized by treewidth admits a polynomial-size approximate Turing kernelization with constant approximation ratio. In the present work, we answer this question affirmatively and in fact provide an efficient polynomial size approximate Turing kernelization scheme (EPSATKS). Moreover, extending the ideas for INDEPENDENT SET, we provide similar results for a variety of other problems.

Our results.   We prove that there is an EPSATKS for a wide variety of graph problems when parameterized by treewidth. The simplest problems we consider are the VERTEX COVER and INDEPENDENT SET problem. Observe that both problems parameterized by treewidth can be shown to be $\mathsf{MK[2]}$-hard, by a simple reduction from CNF-SAT with unbounded clause size.[1] As such, for both problems we indeed do not expect polynomial Turing kernels [22]. We show that VERTEX COVER has a $(1 + \varepsilon)$-approximate Turing kernel with $\mathcal{O}(\frac{\ell}{\varepsilon})$ vertices, and INDEPENDENT SET has a kernel with $\mathcal{O}(\frac{\ell^2}{\varepsilon})$ vertices.

---

[1] A variant of the well-known NP-hardness proof of INDEPENDENT SET (or VERTEX COVER) suffices, where we add two vertices v_x and $v_{\bar{x}}$ for each variable x and connect them. Add a clique for each clause, that has a vertex u_ℓ for each literal ℓ in the clause. Connect u_ℓ to v_x if $\ell = \neg x$, connect u_ℓ to $v_{\bar{x}}$ if $\ell = x$. Observe that the treewidth is bounded by twice the number of variables.

Both approximate Turing kernels follow a similar strategy, based on using separators (originating from the tree decomposition) that separate a piece from the rest of the graph, such that the solution size in this piece is appropriately bounded. For this reason, we formulate a set of conditions on a graph problem and we call graph problems that satisfy these conditions *friendly*. We then show that all friendly graph optimization problems have polynomial-size $(1 + \varepsilon)$-approximate Turing kernels for all $\varepsilon > 0$, when parameterized by treewidth. Precise bounds on the size of the obtained approximate Turing kernels depend on properties of the considered problem, such as the smallest-known (approximate) kernel when parameterized by solution size plus treewidth. In particular, applying the general result for Vertex Cover indeed shows that it has an EPSATKS of size $\mathcal{O}(\frac{\ell}{\varepsilon})$. Using this general technique, we obtain approximate Turing kernels for Clique Cover, Vertex-Disjoint H-Packing for connected graphs H, Feedback Vertex Set, and Edge Dominating Set.

Finally, we prove that Edge Clique Cover and Edge-Disjoint Triangle Packing have an EPSATKS and show that Connected Vertex Cover has a polynomial-size $(1 + \varepsilon)$-approximate Turing kernel. These problems do not satisfy our definition of a friendly problem and require a more problem-specific approach. In particular, for Connected Vertex Cover we will need to consider subconnected tree decompositions [19] and carefully bound the size difference between locally optimal connected vertex covers, and intersections of (global) connected vertex covers with parts of the graph.

Overview. We start in Section 3 by illustrating the general technique using the Vertex Cover problem as an example. We continue by giving the approximate Turing kernels for Edge Clique Cover, Connected Vertex Cover, and Edge-Disjoint Triangle Packing. In Section 4 we state and prove our general theorem and then show that it allows us to give approximate Turing kernels for a number of different graph problems. For statements marked with a ★, the (full) proof can be found in the full version of the paper [24].

2 Preliminaries

We use $\mathbb{N}$ to denote the non-negative integers. Let $[n]$ be defined as the set containing the integers 1 to n. We assume that all graphs are simple and undirected, unless mentioned otherwise. A graph G has vertex set $V(G)$ and edge set $E(G)$. For $v \in V(G)$ we let $d_G(v)$ denote the degree of v. For $X \subseteq V(G)$, we use $G[X]$ to denote the graph induced by vertex set X, we use $G - X$ to denote $G[V(G) \setminus X]$. For $F \subseteq E(G)$, we use $G \setminus F$ to denote the graph resulting from deleting all edges in F from G.

We say that a set $X \subseteq V(G)$ *separates* vertex sets $A \subseteq V(G)$ and $B \subseteq V(G)$ if every path from some vertex in A to some vertex in B contains a vertex in X.

Treewidth. Recall the definition of treewidth.

▶ **Definition 1** ([10]). *A* tree decomposition *of a graph G is a tuple $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree in which each node $t \in V(T)$ has an assigned set of vertices $X_t \subseteq V(G)$, also referred to as the* bag *of node t, such that the following three conditions hold:*
- $\bigcup_{t \in V(T)} X_t = V(G)$, *and*
- *for every edge $\{u, v\} \in E(G)$ there exists $t \in V(T)$ such that $u, v \in X_t$, and*
- *for all $v \in V(G)$ the set $T_v := \{t \in V(T) \mid v \in X_t\}$ induces a connected subtree of T.*

The width *of a tree decomposition of G is the size of its largest bag minus one. The* treewidth *of G is the minimum width of any tree decomposition of G.*

In the remainder of the paper, we will always assume that a tree decomposition is given on input, as treewidth is NP-hard to compute. If it is not, we may use the result below to obtain an approximation of the treewidth and a corresponding tree decomposition in polynomial time. Doing so will weaken any given size bounds in the paper, as it is not a constant-factor approximation. The theorem below is part of [15, Theorem 6.4].

▶ **Theorem 2** ([15, Theorem 6.4]). *There exists a polynomial time algorithm that finds a tree decomposition of width at most $\mathcal{O}(\sqrt{\log tw(G)} \cdot tw(G))$ for a general graph G.*

Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a tree decomposition. Let $t \in V(T)$, we use V_t to denote the set of vertices from G that are contained in some bag of a node in the subtree of T that is rooted at t. It is well-known that for all $t \in V(T)$, the set X_t separates V_t from the rest of the graph. A rooted tree decomposition with root r is said to be *nice* if it satisfies the following properties (cf. [10]).

(i) $X_r = \emptyset$ and $X_t = \emptyset$ for every leaf t of T.

(ii) Every other node is of one of the following three types:

- The node $t \in V(T)$ has exactly two children t_1 and t_2, and $X_t = X_{t_1} = X_{t_2}$. We call such a node a *join* node, or

- the node $t \in V(T)$ has exactly one child t_1, and there exist $v \in V(G)$ such that $X_t = X_{t_1} \cup \{v\}$ (in this case t is an *introduce* node) or such that $X_{t_1} = X_t \cup \{v\}$ (in which case t is a *forget* node).

One can show that a tree decomposition of a graph G of width ℓ can be transformed in polynomial time into a nice tree decomposition of the same width and with $\mathcal{O}(\ell|V(G)|)$ nodes, see for example [10].

To deal with the CONNECTED VERTEX COVER problem we need the tree decomposition to preserve certain connectivity properties. Let a *subconnected tree decomposition* [19] be a tree decomposition where $G[V_t]$ is connected for all $t \in V(T)$. We observe the following.

▶ **Theorem 3** (cf. [19, Theorem 1]). *There is an $\mathcal{O}(n\ell^3)$-algorithm that, given a nice tree decomposition on n nodes of width ℓ of a connected graph G, returns an $\mathcal{O}(n \cdot \ell)$-node subconnected tree decomposition of G, of width at most ℓ such that each node in T has at most $2\ell + 2$ children.*

Proof. Without the additional bound on the degrees of nodes in T, the result is immediate from [19, Theorem 1]. We obtain a subconnected tree decomposition by only executing Phase 1 of Algorithm `make-it-connected` in [19]. Is is shown in [19, Claim 1] that this procedure results in a tree decomposition of width ℓ that is subconnected. It remains to analyze the maximum node degree. The only relevant step of the algorithm is the application of the `split` operation on nodes t from the original tree. Observe that every node in the original tree is visited at most once, and newly introduced nodes are never `split`. If t has parent s, the `split` operation only modifies the degree of s, and any newly introduced nodes. The newly introduced nodes will have degree at most $d_T(t)$. In particular, if s had degree a before the `split` operation on t, it will have degree $a - 1 + p$ after the `split` operation, where p is the number of connected components of $G[V_t]$. We will show that the number of connected components of $G[V_t]$ is bounded by $|X_t|$ if G is a connected graph. We do this by showing that each connected component contains at least one vertex from X_t. Suppose not. Let C be such a component. But since $C \cap X_t = \emptyset$, and X_t is a separator in G, it follows that there are no connections from C to $G[V(G) \setminus V_t]$. If $V_t = V(G)$, then $G[V_t]$ is connected and we are done, otherwise, vertices in $V(G) \setminus V_t$ are not connected to C in G, contradicting that G is connected. Thus, $p \leq |X_t| \leq \ell + 1$. Since in a nice tree decomposition every node has only two children, in the worst case `split` is applied to both these children. Thus, every node in T has degree at most $2\ell + 2$. ◀

Approximation, Kernelization, and Turing Kernelization. We recall the framework for approximate kernelization by Lokshtanov et al. [30] following Fomin et al. [17]. We then introduce suitable definitions for approximate Turing kernelization.

▶ **Definition 4** ([17])**.** *A* parameterized optimization problem $\mathcal{Q}$ *is a computable function*

$$\mathcal{Q} \colon \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}.$$

The instances *of a parameterized optimization problem are pairs* (I, k) *where k is the parameter. A* solution *to* (I, k) *is simply a string* $s \in \Sigma^*$*, such that* $|s| \le |I| + k$*. The* value *of a solution s is given by* $\mathcal{Q}(I, k, s)$*. Using this, we may define the optimal value for the problem as*

$$\mathrm{OPT}_{\mathcal{Q}}(I, k) = \min\{\mathcal{Q}(I, k, s) \mid s \in \Sigma^*, |s| \le |I| + k\},$$

for minimization problems and as

$$\mathrm{OPT}_{\mathcal{Q}}(I, k) = \max\{\mathcal{Q}(I, k, s) \mid s \in \Sigma^*, |s| \le |I| + k\},$$

for maximization problems.

An *optimization problem* $\mathcal{P} \colon \Sigma^* \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$ is defined similarly, but without the parameter. In both cases we will say that s is a *solution* for instance I, if its value is not ∞ (or $-\infty$, in case of maximization problems).

We say that an algorithm for a (regular) minimization problem $\mathcal{P}$ is a *c-approximation algorithm* if for all inputs x it returns a solution s such that the value of s is at most $c \cdot \mathrm{OPT}_{\mathcal{P}}(x)$. Similarly, for a maximization problem we require that s has value at least $\frac{1}{c}\mathrm{OPT}_{\mathcal{P}}(x)$.

When a problem is parameterized by the value of the optimal solution, the definitions of parameterized optimization problems and lossy kernels will cause problems. As such, we use the following interpretation [30, p.229]. Given an optimization problem $\mathcal{P}$ that we want to parameterize by a sum of (potentially multiple) parameters, one of which is the solution value, we define the following corresponding parameterized optimization problem:

$$\mathcal{P}^{\perp}(I, k, s) := \min\{\mathcal{P}(I, s), k + 1\}.$$

In cases where we consider $\mathcal{P}$ parameterized by the treewidth of the input graph, we simply use

$$\mathcal{P}^{\perp}(I, k, s) := \mathcal{P}(I, s).$$

▶ **Definition 5** (α-Approximate kernelization [17])**.** *Let* $\alpha \ge 1$ *be a real number, let g be a computable function and let $\mathcal{Q}$ be a parameterized optimization problem. An* α-approximate kernelization $\mathcal{A}$ *of size g for $\mathcal{Q}$ is a pair of polynomial-time algorithms. The first one is called the* reduction algorithm *and computes a map* $\mathcal{R}_{\mathcal{A}} \colon \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$*. Given as input an instance* (I, k) *of $\mathcal{Q}$, the reduction algorithm computes another instance* $(I', k') = \mathcal{R}_{\mathcal{A}}(I, k)$ *such that* $|I'|, k' \le g(k)$*.*

The second is called the solution-lifting algorithm*. This algorithm takes as input an instance* $(I, k) \in \Sigma^* \times \mathbb{N}$ *of $\mathcal{Q}$, together with* $(I', k') := \mathcal{R}_{\mathcal{A}}(I, k)$ *and a solution s' to* (I', k')*. In time polynomial in* $|I| + |I'| + k + k' + |s|$*, it outputs a solution s to* (I, k) *such that if $\mathcal{Q}$ is a minimization problem, then*

$$\frac{\mathcal{Q}(I, k, s)}{\mathrm{OPT}_{\mathcal{Q}}(I, k)} \le \alpha \cdot \frac{\mathcal{Q}(I', k', s')}{\mathrm{OPT}_{\mathcal{Q}}(I', k')}.$$

For maximization problems we require

$$\frac{\mathcal{Q}(I, k, s)}{\mathrm{OPT}_{\mathcal{Q}}(I, k)} \ge \frac{1}{\alpha} \cdot \frac{\mathcal{Q}(I', k', s')}{\mathrm{OPT}_{\mathcal{Q}}(I', k')}.$$

We say that a problem admits a *Polynomial Size Approximate Kernelization Scheme (PSAKS)* [30] if it admits an α-approximate polynomial kernel for all $\alpha > 1$.

We recall the definition of a Turing kernel, so that we can show how to naturally generalize the notion of approximate kernelization to Turing kernels.

▶ **Definition 6** (Turing kernelization [17])**.** *Let $\mathcal{Q}$ be a parameterized problem and let $f \colon \mathbb{N} \to \mathbb{N}$ be a computable function. A* Turing kernelization *for $\mathcal{Q}$ of size f is an algorithm that decides whether a given instance $(x, k) \in \Sigma^* \times \mathbb{N}$ belongs to $\mathcal{Q}$ in time polynomial in $|x| + k$, when given access to an oracle that decides membership of $\mathcal{Q}$ for any instance (x', k') with $|x'|, k' \leq f(k)$ in a single step.*

In the following definition, we combine the notions of lossy kernelization and Turing kernelization into one, as follows.

▶ **Definition 7** (Approximate Turing kernelization)**.** *Let $\alpha \geq 1$ be a real number, let f be a computable function and let $\mathcal{Q}$ be a parameterized optimization problem. An α-approximate Turing kernel of size f for $\mathcal{Q}$ is an algorithm that, when given access to an oracle that computes a c-approximate solution for instances of $\mathcal{Q}$ in a single step, satisfies the following.*

- *It runs in time polynomial in $|I| + k$, and*
- *given instance (I, k), outputs a solution s such that $\mathcal{Q}(I, k, s) \leq \alpha \cdot c \cdot \mathrm{OPT}_{\mathcal{Q}}(I, k)$ if $\mathcal{Q}$ is a minimization problem and $\mathcal{Q}(I, k, s) \cdot \alpha \cdot c \geq \mathrm{OPT}_{\mathcal{Q}}(I, k)$ is $\mathcal{Q}$ is a minimization problem, and*
- *it only uses oracle-queries of size bounded by $f(k)$.*

Note that, in the definition above, the algorithm does not depend on c, just like in lossy kernelization. We say that a parameterized optimization problem $\mathcal{Q}$ has an *EPSATKS* when it has a polynomial-size $(1 + \varepsilon)$-approximate Turing kernel for every $\varepsilon > 0$, of size $f(\varepsilon) \cdot \mathrm{poly}(k)$ where f is a function that depends only on ε.

3   Approximate Turing kernels for specific problems

In this section we will give approximate Turing kernels for a number of graph problems parameterized by treewidth. We start by discussing the VERTEX COVER problem, since the approximate Turing kernels for all other problems will follow the same overall structure.

3.1   Vertex Cover

In this section we discuss an approximate Turing kernel for VERTEX COVER parameterized by treewidth ℓ. The overall idea will be to use the treewidth decomposition of the graph, and find a subtree rooted at a node t such that $G[V_t \setminus X_t]$ has a large (but not too large) vertex cover. A vertex cover of the entire graph will then be obtained by taking a vertex cover of $G[V_t \setminus X_t]$, adding all vertices in X_t, and recursing on the graph that remains after removing V_t. This produces a correct vertex cover because X_t is a separator in the graph. Furthermore, taking all of X_t into the vertex cover is not problematic as X_t is ensured to be comparatively small. To obtain a vertex cover of $G[V_t \setminus X_t]$, we will use the following lemma.

▶ **Lemma 8** (★)**.** *Let G be a graph with $OPT_{VC}(G) \leq k$. Then there is a polynomial-time algorithm returning vertex cover of G of size at most $c \cdot \mathrm{OPT}_{VC}(G)$, when given access to c-approximate oracle that solves vertex cover on graphs with at most $\mathcal{O}(k)$ vertices.*

Proof sketch. First apply the LP-based kernel [9] for VERTEX COVER parameterized by solution size to (G, k). This gives an instance G' with at most $2k$ vertices. We can then apply the oracle to obtain a c-approximate vertex cover of G'. We show in the complete proof that it is straightforward to lift this solution to a c-approximate vertex cover of the original graph G, concluding the proof. ◀

Using this, we can now give the $(1 + \varepsilon)$-approximate Turing kernel for VERTEX COVER. While the theorem statement requires $\varepsilon \leq 1$, this does not really impose a restriction: if $\varepsilon > 1$ one may simply reset it to be 1. It simply shows that the bounds do not continue improving indefinitely as ε grows larger than 1. Note however that VERTEX COVER is 2-approximable in polynomial time, such that choosing ε larger than one is likely not useful.

▶ **Theorem 9.** *For every $0 < \varepsilon \leq 1$, VERTEX COVER parameterized by treewidth ℓ has a $(1 + \varepsilon)$-approximate Turing kernel with $\mathcal{O}(\frac{\ell}{\varepsilon})$ vertices.*

Proof. Consider Algorithm 1, we use the well-known 2-approximation algorithm for VERTEX COVER. First of all, we show how to do Step 8 of the algorithm efficiently.

Algorithm 1 An approximate Turing kernel for VERTEX COVER.

1: **procedure** APPROXVC$(G, \mathcal{T}, \varepsilon)$
2:     Turn $\mathcal{T}$ into a nice tree decomposition of G
3:     Obtain a 2-approximate solution $\tilde{S}$ for VC in G
4:     **if** $|\tilde{S}| \leq \frac{8(\ell+1)}{\varepsilon}$ **then**
5:         Determine a c-approximate solution S to VC in G using Lemma 8
6:         **return** S
7:     **else**
8:         Find $t \in V(T)$ s.t. $\frac{(\ell+1)}{\varepsilon} \leq \text{OPT}_{\text{VC}}(G[V_t \setminus X_t]) \leq \frac{8(\ell+1)}{\varepsilon}$
9:         Determine a c-approximate solution S_t to VC in $G[V_t \setminus X_t]$ using Lemma 8
10:        $G' \leftarrow G - V_t$
11:        Let $\mathcal{T}'$ be $\mathcal{T}$ after removing the subtree rooted at t and all vertices in X_t
12:        $S' \leftarrow$ APPROXVC$(G', \mathcal{T}', \varepsilon)$
13:        **return** $S_t \cup X_t \cup S'$
14:    **end if**
15: **end procedure**

▷ **Claim.** There is a polynomial-time algorithm that, given graph G such that $\text{OPT}_{\text{VC}}(G) \geq \frac{(\ell+1)}{\varepsilon}$, with a nice tree decomposition $\mathcal{T}$ of width at most ℓ, outputs a node $t \in V(T)$ such that $\frac{(\ell+1)}{\varepsilon} \leq \text{OPT}_{\text{VC}}(G[V_t \setminus X_t]) \leq \frac{8(\ell+1)}{\varepsilon}$.

Proof. Let $\mathcal{T}$ be a nice tree decomposition with root r. We start from $t := r$, maintaining that $\text{OPT}_{\text{VC}}(G[V_t \setminus X_t]) \geq \frac{\ell+1}{\varepsilon}$. Note that this is initially true since $G_r = G$.

Check whether the 2-approximation returns a vertex cover of size at most $\frac{8(\ell+1)}{\varepsilon}$ for $G[V_t \setminus X_t]$. If yes, we are done. If not, then $\text{OPT}_{\text{VC}}(G[V_t \setminus X_t]) > \frac{4(\ell+1)}{\varepsilon}$. We show that t has a child on which we will recurse. We do a case distinction on the type of node of t.

- t is a leaf node. In this case, $|V_t \setminus X_t| = 0$, contradicting that $\text{OPT}_{\text{VC}}(G[V_t \setminus X_t]) > \frac{4(\ell+1)}{\varepsilon} \geq 0$.
- t is a forget or introduce node. This implies t has one child t_1 and the size of $V_t \setminus X_t$ and $V_{t_1} \setminus X_{t_1}$ differs by at most one. Therefore, $\text{OPT}_{\text{VC}}(G[V_{t_1} \setminus X_{t_1}]) \geq \text{OPT}_{\text{VC}}(G[V_t \setminus X_t]) - 1 \geq \text{OPT}_{\text{VC}}(G[V_t \setminus X_t])/2$.

- t is a join node with children t_1 and t_2. Observe that $G[V_t \setminus X_t]$ is the disjoint union of the graphs $G[V_{t_1} \setminus X_{t_1}]$ and $G[V_{t_2} \setminus X_{t_2}]$ (note $X_t = X_{t_1} = X_{t_2}$). As such, for one of the two children, without loss of generality let this be t_1, running the 2-approximation algorithm for vertex cover returns a value of at least $\text{OPT}_{\text{VC}}(G[V_t \setminus X_t])/2$, meaning that $\text{OPT}_{\text{VC}}(G[V_{t_1} \setminus X_{t_1}]) \geq \text{OPT}_{\text{VC}}(G[V_t \setminus X_t])/4$.

Thus, there is a child t_1 such that $\text{OPT}_{\text{VC}}(G[V_{t_1} \setminus X_{t_1}]) \geq \text{OPT}_{\text{VC}}(G[V_t \setminus X_t])/4 \geq \frac{\ell+1}{\varepsilon}$. Continue with $t := t_1$. ◁

We will now show the correctness of the algorithm by induction on $|V(G)|$. Let G be a graph with nice tree decomposition $\mathcal{T}$. If the algorithm returns a VERTEX COVER in Step 5, the result is immediate. If not, then it follows that the algorithm returns in Step 13, and that $\text{OPT}_{\text{VC}}(G) > \frac{4(\ell+1)}{\varepsilon}$. The algorithm then returns a vertex cover S_t for $G[V_t \setminus X_t]$ together with X_t and a vertex cover $S' = \text{APPROXVC}(G', \mathcal{T}', \varepsilon)$ in the remainder of the graph. It is easy to see that the returned set is indeed a vertex cover of the graph. Furthermore, one may verify that the oracle is only used for graphs with at most $\mathcal{O}(\frac{\ell}{\varepsilon})$ vertices. It remains to verify the approximation ratio. Recall that $G' := G - V_t$. Then

$$|S_t| + |S'| + |X_t| \leq c \cdot \text{OPT}_{\text{VC}}(G[V_t \setminus X_t]) + c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{VC}}(G') + \ell + 1$$
$$\leq c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{VC}}(G[V_t \setminus X_t]) + c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{VC}}(G')$$
$$\leq c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{VC}}(G). \qquad \blacktriangleleft$$

3.2 Edge Clique Cover

In this section, we obtain an approximate Turing kernel for EDGE CLIQUE COVER, which is defined as follows.

EDGE CLIQUE COVER (ECC)   **Parameter:** ℓ

Input: A graph G with tree decomposition $\mathcal{T}$ of width ℓ.

Output: The minimum value for $k \in \mathbb{N}$ such that there exists a family S of subsets of $V(G)$ such that $|S| \leq k$, $G[C]$ is a clique for all $C \in S$, and for all $\{u,v\} \in E(G)$ there exists $C \in S$ such that $u, v \in S$?

To obtain an approximate Turing kernel, we will separate suitably-sized subtrees from the graph using the tree decomposition, as we did in the approximate Turing kernel for VERTEX COVER. To show that this results in the desired approximation bound, we will need the following lemma. It basically shows that if we find a node t of the tree decomposition such that X_t is "small" compared to $\text{OPT}(V_t)$, we will be able to combine an edge clique cover in $G[V_t]$ with one in $G - (V_t \setminus X_t)$ to obtain a clique cover of the entire graph that is not too far from optimal.

▶ **Lemma 10.** *Let G be a graph, let $X_1, X_2 \subseteq V(G)$ such that $X_1 \cup X_2 = V(G)$ and $X = X_1 \cap X_2$ separates X_1 from X_2 in G. Then*

$$OPT_{ECC}(G) \geq OPT_{ECC}(G[X_1]) + OPT_{ECC}(G[X_2]) - \binom{|X|}{2}.$$

Proof. Let S be an edge clique cover of G. We show how to obtain clique covers S_1 and S_2 for $G[X_1]$ and $G[X_2]$ such that $|S_1| + |S_2| \leq |S| + \binom{|X|}{2}$. First define

$$S_1' := \{C \mid C \cap (X_1 \setminus X) \neq \emptyset, C \in S\} \cup \{C \mid C \subseteq X, C \in S\},$$

similarly, define

$$S_2' := \{C \mid C \cap (X_2 \setminus X) \neq \emptyset, C \in S\}.$$

For $j \in [2]$, define $S_j := S_j' \cup S_j''$, where $S_j'' := \{\{u, v\} \in E(G[X]) \mid \{u, v\}$ not covered by $S_j'\}$.

We start by showing that S_j is an edge clique cover of $G[X_j]$ for $i \in [2]$. First of all, we will verify that $C \subseteq X_j$ and that C forms a clique in $G[X_j]$ for all $C \in S_j$. For $C \in S_j''$ this is trivial, for $C \in S_j'$, observe that C is a clique in G and any clique in G containing a vertex from $X_j \setminus X$ cannot contain a vertex from $V(G) \setminus X_j$, since X is a separator. Thus $C \subseteq X_j$. The fact that C is a clique in $G[X_j]$ is immediate from C being a clique in G.

It remains to show that S_j covers all edges in $G[X_j]$. Let $\{u, v\} \in E(G[X_j])$. If $u, v \in X$, then the edge is covered by definition. Without loss of generality, suppose $u \in X_j \setminus X$. Let $C \in S$ be a clique that covered edge $\{u, v\}$. Then clearly $u \in C \cap (X_j \setminus X)$ and thus $C \cap (X_j \setminus X) \neq \emptyset$, implying $C \in S_j$. Thus, the edge $\{u, v\}$ is indeed covered by S_j.

It remains to show that $|S_1| + |S_2| \leq |S| + \binom{|X|}{2}$. Start by observing that $|S_1'| + |S_2'| \leq |S|$, since a clique cannot contain both a vertex from $X_1 \setminus X$ and $X_2 \setminus X$. Since every edge $\{u, v\} \in E(G[X])$ is covered by S, it is easy to observe from the definition that $\{u, v\}$ is covered by S_1' or S_2'. As such, $S_1'' \cap S_2'' = \emptyset$. Since $G[X]$ has at most $\binom{|X|}{2}$ edges, it follows that $|S_1''| + |S_2''| \leq \binom{|X|}{2}$ and indeed $|S_1| + |S_2| \leq |S_1'| + |S_1''| + |S_2'| + |S_2''| \leq |S| + \binom{|X|}{2}$.   ◄

Before giving the approximate Turing kernel, we show that there exists a node t in the tree decomposition such that the size of the subtree rooted at t falls within certain size bounds. We use this to split off subtrees, similar to the strategy we used for VERTEX COVER earlier.

▶ **Lemma 11 (★).** *There is a polynomial-time algorithm that, given a graph G with $|V(G)| \geq 2\frac{1+\varepsilon}{\varepsilon}(\ell+1)^4$, a nice tree decomposition $\mathcal{T}$ of width ℓ, and $\varepsilon > 0$, outputs a node $t \in V(T)$ such that $2\frac{1+\varepsilon}{\varepsilon}(\ell+1)^4 \leq |V_t \setminus X_t| \leq 4\frac{1+\varepsilon}{\varepsilon}(\ell+1)^4$.*

Using the lemma above, we can now give the approximate Turing kernel for EDGE CLIQUE COVER.

▶ **Theorem 12.** *For every $0 < \varepsilon \leq 1$, EDGE CLIQUE COVER parameterized by treewidth ℓ has a $(1+\varepsilon)$-approximate Turing kernel with $\mathcal{O}(\frac{\ell^4}{\varepsilon})$ vertices.*

Proof. Consider Algorithm 2, we show that it is a $(1+\varepsilon)$-approximate Turing kernel for ECC. Observe that Step 2 can be done efficiently while maintaining a valid tree decomposition, as one may simply restrict the bags of the decomposition to the relevant connected component of G. It is easy to verify that the procedure runs in polynomial time, using that $|V_t \setminus X_t|$ is always non-empty and thus the recursive call is on a strictly smaller graph. Finally, we can verify the size-bound, as the oracle is only applied to G if $|V(G)| \leq \mathcal{O}(\frac{\ell^4}{\varepsilon})$ or to $G[V_t]$ when $|V_t \setminus X_t| \leq \mathcal{O}(\frac{\ell^4}{\varepsilon})$, implying that $|V_t| \leq |V_t \setminus X_t| + \ell + 1 = \mathcal{O}(\frac{\ell^4}{\varepsilon})$.

We continue by showing that Algorithm 2 returns an edge clique cover of G. If the algorithm returns in Step 6, this is immediate. Otherwise, observe that since X_t separates V_t and $V(G')$ in G, it follows that any edge in G is in $E(G[V_t])$ or in $E(G')$. Thus, such an edge is covered by S_t or S', implying that $S = S_t \cup S'$ is an edge clique cover of G. We now bound $|S_t| + |S'|$, to show that the algorithm indeed approximates the optimum ECC.

$$\begin{aligned} |S_t| + |S'| &\leq c \cdot \mathrm{OPT}_{\mathrm{ECC}}(G[V_t]) + |S'| \\ &= c \cdot (1+\varepsilon) \cdot \mathrm{OPT}_{\mathrm{ECC}}(G[V_t]) - c \cdot \varepsilon \cdot \mathrm{OPT}_{\mathrm{ECC}}(G[V_t]) + |S'| \end{aligned}$$

■ **Algorithm 2** An approximate Turing Kernel for EDGE CLIQUE COVER.

1: **procedure** APPROXECC$(G, \mathcal{T}, \varepsilon)$
2:     If G is not connected, split G into its connected components and treat them separately.
3:     Turn $\mathcal{T}$ into a nice tree decomposition.
4:     **if** $|V(G)| \leq \frac{2(1+\varepsilon)}{\varepsilon}(\ell+1)^4$ **then**
5:         Apply the c-approximate oracle to obtain an ECC S of G
6:         **return** S
7:     **else**
8:         Find $t \in V(T)$ s.t. $2\frac{(1+\varepsilon)}{\varepsilon}(\ell+1)^4 \leq |V_t \setminus X_t| \leq \frac{4(1+\varepsilon)}{\varepsilon}(\ell+1)^4$ (by Lemma 11)
9:         Determine a c-approximate solution S_t to ECC in $G[V_t]$ using the oracle
10:        $G' \leftarrow G - (V_t \setminus X_t)$
11:        Let $\mathcal{T}'$ be $\mathcal{T}$ after removing the subtree rooted at t except for t
12:        $S' \leftarrow$ APPROXECC$(G', \mathcal{T}', \varepsilon)$
13:        **return** $S_t \cup S'$
14:    **end if**
15: **end procedure**

Observe that every clique covers at most $\binom{\ell+1}{2}$ edges, since it has at most $\ell+1$ vertices, since the treewidth of G is bounded by ℓ. Thus $\text{OPT}_{\text{ECC}}(G[V_t]) \geq |E(G[V_t])|/\binom{\ell+1}{2}$.

$$\leq c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{ECC}}(G[V_t]) - c \cdot \varepsilon \cdot |E(G[V_t])|/\binom{\ell+1}{2} + |S'|$$

Observe that $V_t \setminus X_t$ cannot contain vertices that are isolated in $G[V_t]$, since G is connected and X_t separates V_t from the remainder of G. Thus, $|E(G[V_t])| \geq |V_t \setminus X_t|/2$.

$$\leq c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{ECC}}(G[V_t]) - c \cdot \varepsilon \cdot \frac{|V_t \setminus X_t|}{2(\ell+1)^2} + |S'|$$

$$\leq c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{ECC}}(G[V_t]) - c \cdot (1+\varepsilon) \cdot (\ell+1)^2 + |S'|$$

using $\ell+1 \geq |X_t|$

$$\leq c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{ECC}}(G[V_t]) - c \cdot (1+\varepsilon) \cdot \binom{|X_t|}{2} + |S'|$$

$$\leq c \cdot (1+\varepsilon) \cdot \left(\text{OPT}_{\text{ECC}}(G[V_t]) + \text{OPT}_{\text{ECC}}(G') - \binom{|X_t|}{2}\right)$$

By Lemma 10

$$\leq c \cdot (1+\varepsilon) \cdot \text{OPT}_{\text{ECC}}(G). \qquad \blacktriangleleft$$

3.3 Edge-Disjoint Triangle Packing

In this section we give an approximate Turing kernel for the EDGE-DISJOINT TRIANGLE PACKING problem, defined as follows.

EDGE-DISJOINT TRIANGLE PACKING (ETP)   **Parameter:** ℓ

Input: A graph G with tree decomposition $\mathcal{T}$ of width ℓ.

Output: The maximum value for $k \in \mathbb{N}$ such that there exists a family S of size-3 subsets of $V(G)$ such that $|S| \geq k$, $G[X]$ is a triangle for all $X \in S$, and X and Y are edge-disjoint for all $X, Y \in S$?

Observe that the problem has a 3-approximation by taking any maximal edge-disjoint triangle packing S, which can be greedily constructed. This packing then uses $3|S|$ edges. If there is a solution S' with $|S'| > 3|S|$, then there is a triangle in S' that contains no edge covered by S, contradicting that S is maximal. We now give the approximate Turing kernel.

▶ **Theorem 13.** *For every $0 \leq \varepsilon \leq 1$, EDGE-DISJOINT TRIANGLE PACKING parameterized by treewidth ℓ, has a $(1 + \varepsilon)$-approximate Turing kernel with $\mathcal{O}(\frac{\ell^2}{\varepsilon})$ vertices.*

Proof. We will use the following claim.

▷ **Claim 14 (★).** Let G be a graph with $\mathrm{OPT}_{\mathrm{ETP}}(G) \leq k$. There is a polynomial-time algorithm that when given access to a c-approximate oracle, outputs a c-approximate solution for G using calls to the oracle with at most $\mathcal{O}(k)$ vertices.

We now describe the algorithm. Start by computing a 3-approximate solution $\tilde{S}$ to EDGE-DISJOINT TRIANGLE PACKING in G. If $|\tilde{S}| \leq 18\frac{(\ell+1)^2}{\varepsilon}$, we obtain an approximate solution to triangle packing using Claim 14.

Otherwise, for $t \in V(T)$ define G_t as $G[V_t] \setminus E(G[X_t])$, i.e., the graph $G[V_t]$ from which the edges between vertices in X_t have been removed. We show how to find $t \in T$ such that

$$\frac{(\ell+1)^2}{\varepsilon} \leq \mathrm{OPT}_{\mathrm{ETP}}(G_t) \leq 18\frac{(\ell+1)^2}{\varepsilon},$$

together with an approximate solution S_t in G_t. Start with $t := r$, observe that initially $\mathrm{OPT}_{\mathrm{ETP}}(G_t) > \frac{18(\ell+1)^2}{\varepsilon}$ since $G_r = G$ and $\mathrm{OPT}_{\mathrm{ETP}}(G_t) \geq |\tilde{S}|$. So suppose we are at some node t with $\mathrm{OPT}_{\mathrm{ETP}}(G_t) \geq \frac{(\ell+1)^2}{\varepsilon}$. Compute a 3-approximate solution in G_t. If this solution has value at most $\frac{6(\ell+1)^2}{\varepsilon}$, we obtain an approximate solution S_t to triangle packing in G_t using Claim 14. Otherwise, we will recurse on a child t_1 of t for which $\mathrm{OPT}_{\mathrm{ETP}}(G_{t_1}) \geq \frac{(\ell+1)^2}{\varepsilon}$, we show how to find such a child by doing a case distinction on the type of node of t.

- t is a leaf node. This is a contradiction with the assumption that $\mathrm{OPT}_{\mathrm{ETP}}(G_t) > 6\frac{(\ell+1)^2}{\varepsilon}$, since G_t is empty.
- t has exactly one child t_1 and $X_t = X_{t_1} \cup \{v\}$ for some $v \in V(G)$. This means in particular that $G_{t_1} = G_t - \{v\}$. Furthermore, we can show that v is isolated in G_t. After all, there are no edges between vertices in X_t and v by definition of G_t. Furthermore, there are no edges between v and vertices not in X_t, by correctness of the tree decomposition. Therefore, trivially, $\mathrm{OPT}_{\mathrm{ETP}}(G_t) = \mathrm{OPT}_{\mathrm{ETP}}(G_{t_1})$ and we continue with $t \leftarrow t_1$.
- t has exactly one child t_1 and $X_t = X_{t_1} \setminus \{v\}$ for some $v \in V(G)$. In this case G_{t_1} can be obtained by G_t by removing all edges between vertices in v and vertices in X_t. This removes at most $(\ell+1)$ edges from the graph, and thus $\mathrm{OPT}_{\mathrm{ETP}}(G_{t_1}) \geq \mathrm{OPT}_{\mathrm{ETP}}(G_t) - \ell \geq \frac{(\ell+1)^2}{\varepsilon}$, and we continue with $t \leftarrow t_1$.
- t is a join node with children t_1 and t_2. Observe that X_t separates G_t and that $\mathrm{OPT}_{\mathrm{ETP}}(G_t) = \mathrm{OPT}_{\mathrm{ETP}}(G_{t_1}) + \mathrm{OPT}_{\mathrm{ETP}}(G_{t_2})$. As such, there is a child of G_t, w.l.o.g. let this be t_1, such that $\mathrm{OPT}_{\mathrm{ETP}}(G_{t_1}) \geq \mathrm{OPT}_{\mathrm{ETP}}(G_t)/2 \geq \frac{3(\ell+1)^2}{\varepsilon}$. Using the 3-approximation on both children, find a child where the returned solution size is at least $\frac{3(\ell+1)^2}{3\varepsilon} = \frac{(\ell+1)^2}{\varepsilon}$. Continue with this child.

Using t and the obtained solution S_t in G_t, we now do the following. Let $G' := G - (V_t \setminus X_t)$. Obtain a solution S' in G' using the algorithm above on the smaller graph G'. Output $S := S_t \cup S'$. Since G' and G_t are edge-disjoint subgraphs of G, it is easy to observe that S is an edge-disjoint triangle packing in G.

It remains to show that S has the desired size. Observe that the size of an edge-disjoint triangle packing in G can be bounded by considering the triangles whose edges are in G_t, those whose edges are in G', and those with at least one edge with both endpoints in X_t. Using that there are at most $\binom{X_t}{2}$ edges between vertices in X_t, we get

$$\mathrm{OPT}_{\mathrm{ETP}}(G) \leq \mathrm{OPT}_{\mathrm{ETP}}(G_t) + \mathrm{OPT}_{\mathrm{ETP}}(G') + \binom{X_t}{2}$$

$$\leq (1 + \varepsilon)\mathrm{OPT}_{\mathrm{ETP}}(G_t) + \mathrm{OPT}_{\mathrm{ETP}}(G')$$
$$\leq c \cdot (1 + \varepsilon)|S_t| + c \cdot (1 + \varepsilon)|S'|$$
$$\leq c \cdot (1 + \varepsilon)|S|. \hspace{2cm} \blacktriangleleft$$

The strategy used to obtain a kernel for EDGE-DISJOINT TRIANGLE PACKING can be generalized to packing larger cliques, as long as these problems have polynomial kernels when parameterized by solution size. Generalizing to the more general question of packing edge-disjoint copies of some other graph H may be more difficult. In this case, there can be copies of H that have vertices in both sides of the graph after removing the edges within a separator, and one needs to be careful to not discard too many of these.

3.4   Connected Vertex Cover

The CONNECTED VERTEX COVER (CVC) problem asks, given a graph G and tree decomposition $\mathcal{T}$, for the minimum size of a vertex cover S in G such that $G[S]$ is connected. It is known that CVC has a $(1 + \varepsilon)$-approximate kernel of polynomial size [30].

▶ **Theorem 15** ([30]). *CONNECTED VERTEX COVER parameterized by solution size k admits a strict time efficient PSAKS with $\mathcal{O}(k^{\lceil \frac{\alpha}{\alpha-1} \rceil} + k^2)$ vertices.*

To obtain an approximate Turing kernel, we will use a similar strategy to the Turing kernel for VERTEX COVER described in Theorem 9. However, the connectivity constraint makes this kernel somewhat more complicated. We deal with this by changing the procedure in two places. First of all, we will use a subconnected tree decomposition, to ensure that $G[V_t]$ is connected for any node t. We will then again find a subtree with a suitably-sized solution. In this case however, we will contract the separator between the subtree and the rest of the graph to a single vertex. The next lemma shows that this does not reduce the connected vertex cover size in the subtree by more than twice the size of the separator.

▶ **Lemma 16.** *Let G be a connected graph and let $X \subseteq V(G)$. Given a connected vertex cover S of G_X where G_X is obtained from X by identifying all vertices from X into a single vertex z, there is a polynomial-time algorithm that finds a connected vertex cover S' of size at most $|S| + 2|X|$ of G.*

Proof. Let S be a connected vertex cover of G_X. Let $S'' := S \cup X \setminus \{z\}$. Observe that S'' is a vertex cover of G, such that every connected component of $G[S'']$ contains at least one vertex from X; thus, there are at most $|X|$ connected components. If $G[S'']$ is connected, we are done. Otherwise, we show that there is a single vertex $v \in V(G)$ such that $G[S'' \cup \{v\}]$ has strictly fewer connected components than $G[S'']$. It is then straightforward to obtain S' by repeatedly adding such a vertex, until $G[S'']$ is connected. For any vertex $u \in S''$ define C_u as the connected component of vertex u in $G[S'']$.

Let x and x' be in two distinct components in $G[S'']$, consider the shortest path P from x to x' in G. Refer to Figure 1 for a sketch of the situation. By this definition, $C_x \neq C_{x'}$. Let y be the first vertex in P such that $y \in S''$ but $y \notin C_x$, let y' be the vertex on P before
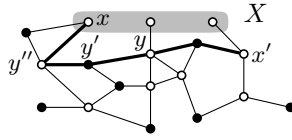
Figure 1 A graph with a vertex cover S'' (indicated in white) that is connected when all vertices in X are identified into a single vertex. Shown are x, x',y,y',y'', and P (indicated in bold) as used in the proof of Lemma 16.

y, observe that $y' \notin S''$ since otherwise $y' \in S''$ and $y' \notin C_x$ which is a contradiction with the fact that y is the first such vertex in P. Let y'' be the vertex on the path before y', such that $P = (x, \ldots, y'', y', y, \ldots, x')$, where possibly $x = y''$ or $y = x'$. Observe that $y'' \in S''$ as otherwise edge $\{y'', y'\}$ is not covered, and therefore $y'' \in C_x$ since y is the first vertex on P that is in S'' but not in C_x. Therefore, adding vertex y' to S'' will merge connected components C_x and C_y, such that the number of connected components in $G[S'' \cup \{y'\}]$ is strictly smaller than the number of connected components in $G[S'']$. In total, we add less than $|X|$ vertices to S'' obtain a connected vertex cover S' and thus $|S'| \leq |S| + |X|$.   ◄

We now prove the main result of this section.

▶ **Theorem 17.** *For every $0 < \varepsilon \leq 1$, Connected Vertex Cover parameterized by treewidth ℓ has a $(1 + \varepsilon)$-approximate Turing Kernel with $\mathcal{O}((\frac{\ell^2}{\varepsilon})^{\lceil \frac{3+\varepsilon}{\varepsilon} \rceil})$ vertices.*

Proof. We will use the PSAKS for Connected Vertex Cover from Theorem 15. Recall that such a PSAKS consists of a reduction algorithm $R_{\mathcal{A}}$ together with a solution lifting algorithm $S_{\mathcal{A}}$. We will use the following claim.

▷ **Claim 18.** Given $0 < \delta \leq 1$ and a connected graph G with tree decomposition of width ℓ, there is a polynomial-time algorithm to determine a d-approximate solution for CVC or correctly decide that $\text{OPT}_{\text{CVC}}(G) > \frac{100\ell^2}{\delta}$, when given access to a c-approximate CVC-oracle that allows calls using graphs with at most $\mathcal{O}((\frac{\ell^2}{\delta})^{\lceil \frac{1+\delta}{\delta} \rceil})$ vertices, where $d = \min(c \cdot (1+\delta), 2)$.

Proof. Using the fact that CVC is 2-approximable in polynomial time [33], obtain a 2-approximate solution $\tilde{S}$ in G. If $|\tilde{S}| > 200\ell^2/\delta$, return NO and halt. Otherwise, continue by running $R_{\mathcal{A}}$ on $(G, |\tilde{S}|)$ to obtain (G', k'). Observe that G' has at most $\mathcal{O}((\frac{\ell^2}{\delta})^{\lceil \frac{1+\delta}{\delta} \rceil})$ many vertices. Apply the c-approximate oracle on G' to obtain CVC S' in G'. Obtain an approximate solution S in G by using the solution lifting algorithm on G' and S'. Output the smallest solution of S and $\tilde{S}$, let this be $\hat{S}$. We show that this has the desired approximation factor, which requires an argument since the PSAKS works for $\text{CVC}^\perp$ instead of CVC (recall $\text{CVC}^\perp(G, k, S) = \min\{k + 1, \text{CVC}(G, S)\}$). Observe that $|\hat{S}| \leq |\tilde{S}|$, by definition. Therefore, $|\hat{S}| \leq \text{CVC}^\perp(G, |\tilde{S}|, S)$. Thus

$$\frac{|\hat{S}|}{\text{OPT}_{\text{CVC}}(G)} \leq \frac{\text{CVC}^\perp(G, |\tilde{S}|, S)}{\text{OPT}_{\text{CVC}}(G)} \leq \frac{\text{CVC}^\perp(G, |\tilde{S}|, S)}{\text{OPT}_{\text{CVC}^\perp}(G, |\tilde{S}|)}.$$

By correctness of the solution lifting algorithm, we get

$$\frac{\text{CVC}^\perp(G, |\tilde{S}|, S)}{\text{OPT}_{\text{CVC}^\perp}(G, |\tilde{S}|)} \leq (1+\delta) \frac{\text{CVC}^\perp(G', k', S')}{\text{OPT}_{\text{CVC}^\perp}(G', k')} \leq (1+\delta) \frac{|S'|}{\text{OPT}_{\text{CVC}}(G')} \leq c \cdot (1+\delta),$$

by correctness of the oracle.   ◁

Algorithm. The algorithm now proceeds as follows. Our goal is to find a subtree of T for which on the one hand, the local optimum CVC is small enough to find an approximate solution using Claim 18, but also large enough to be able to (among other things) add the entire set X_t to the solution, without introducing a too large error. Let $\delta := \varepsilon/3$.

For any vertex $t \in V(T)$, let G_t be the graph given by $G[V_t]$ after identifying all vertices from X_t into a single vertex z_t. Apply Claim 18 to G, if it returns an approximate connected vertex cover of G, we are done. Otherwise, $\text{OPT}_{\text{CVC}}(G) > \frac{100\ell^2}{\delta}$. We now aim to find a vertex t such that Claim 18 returns an approximate solution in G_t of size at least $\frac{10\ell}{\delta}$.

$\triangleright$ **Claim 19.** There is a polynomial-time algorithm that, given G with tree decomposition $\mathcal{T}$ of width ℓ such that $\text{OPT}_{\text{CVC}}(G) > \frac{100\ell^2}{\delta}$, finds $t \in V(T)$ for which Claim 18 returns an approximate solution S_t with $|S_t| \geq \frac{10\ell}{\delta}$, using calls to a c-approximate oracle of size at most $\mathcal{O}((\frac{\ell^2}{\delta})^{\lceil \frac{1+\delta}{\delta} \rceil})$.

Proof. Start with $t := r$, note that since $\text{OPT}_{\text{CVC}}(G) > \frac{100\ell^2}{\delta}$ and $G_r = G$, we have that $\text{OPT}_{\text{CVC}}(G_r) > \frac{100\ell^2}{\delta}$, where r is the root of T. We search through the graph maintaining $\text{OPT}_{\text{CVC}}(G_t) > \frac{100\ell^2}{\delta}$. Let $t_1, \ldots, t_m$ be the children of t, recall that we may assume $m \leq 2\ell + 2$ by Theorem 3. For each t_i, apply Claim 18. Consider the following possibilities.

- There exists $i \in [m]$ such that the claim determines $\text{OPT}_{\text{CVC}}(G_{t_i}) > \frac{100\ell^2}{\delta}$, in this case, recurse with this t_i.
- There exists $i \in [m]$ such that the claim returns a $\min\{2, (1+\delta) \cdot c\}$-approximate solution S_{t_i} of size at least $\frac{10\ell}{\delta}$ for CVC. In this case, return $t := t_i$.
- Otherwise. Thus, for every $i \in [m]$, the algorithm returns a connected vertex cover S_i of size at most $\frac{10\ell}{\delta}$ for CVC in G_{t_i}. Obtain a connected vertex cover S_i' of $G[V_{t_i}]$ of size at most $|S_i| + 2(\ell+1)$ using Lemma 16. We will argue that in this case $CVC(G_t) < \frac{55\ell^2}{\delta}$, which is a contradiction. We obtain a connected vertex cover of G_t as follows. Let $\hat{S}_t := \bigcup_{i \in [m]}(S_i') \cup \{z_t\}$. Observe that $\hat{S}_t$ has size at most $(2\ell + 2) \cdot \frac{13\ell}{\delta} + 1 \leq \frac{55\ell^2}{\delta}$. It is easy to observe that $\hat{S}_t$ is indeed a connected vertex cover of G_t.

Observe that from the steps above, we always get a connected vertex cover S_t of G_t, that is a $(1+\delta) \cdot c$-approximation of $\text{OPT}_{\text{CVC}}(G_t)$ and has size at least $\frac{10\ell}{\delta}$. $\triangleleft$

Using Claim 19, we obtain a node t and a connected vertex cover S_t of G_t, that is a $\min\{(1+\delta) \cdot c, 2\}$-approximation of $\text{OPT}_{\text{CVC}}(G_t)$ and has size at least $\frac{10\ell}{\delta}$. Use Lemma 16 to obtain a connected vertex cover S_t' of $G[V_t]$ of size at most $|S_t| + 2(\ell+1)$, containing X_t.

We now obtain graph G' by removing all vertices in $V_t \setminus X_t$ from G and then contracting all vertices in X_t to a single vertex z_t. Let $\mathcal{T}'$ to be a tree decomposition of G', one may obtain $\mathcal{T}'$ by replacing occurrences of vertices in V_t by z_t in $\mathcal{T}$. Since G' is strictly smaller than G, we may use the algorithm described above to obtain a $c \cdot (1+\varepsilon)$-approximate solution S' for $\text{OPT}_{\text{CVC}}(G')$, using $\mathcal{T}'$. Output $S := S' \cup S_t' \setminus \{z_t\}$.

Correctness. We start by showing that S is a connected vertex cover. Verify that it is indeed a vertex cover of G: any edge within G' is covered as $S' \subseteq S$, any edge in G_t is covered since $S_t' \subseteq S$ and any other edge has at least one endpoint in $X_t \subseteq S$ and is thereby covered. It remains to verify that $G[S]$ is connected. Clearly, $G[V_t \cap S]$ is connected since it corresponds to $G[S_t']$. Let $\tilde{G} := G - (V_t \setminus X_t)$. We show that every connected component of $\tilde{G}[S]$ contains at least one vertex from X_t, such that the entire graph is connected as $X_t \subseteq S$ and the vertices in X_t are in the same connected component as observed earlier. Suppose not, let C be such a component not containing any vertex in X_t. Consider $G'[S']$. Observe that C is also a connected component of $G'[S']$. Furthermore, vertex z_t is not adjacent to

any vertex in C, as otherwise there is an edge from some vertex in C to some vertex in X_t in $\tilde{G}$, since $X_t \subseteq S$ this contradicts that C contains no vertex from X_t. Since G' is connected however, z_t has an incident edge $\{z_t, u\}$ for some $u \in V(G')$ and thus $u \in S'$ or $z_t \in S'$. In both cases there is a vertex in S' that is not in connected component C, a contradiction with the assumption that S' is a connected vertex cover of G'.

We now show that we indeed achieve the desired approximation factor.

▷ **Claim 20.** $|S| \leq c \cdot (1 + \varepsilon) \cdot \mathrm{OPT}_{\mathrm{CVC}}(G)$

Proof. Let S^* be a minimum connected vertex cover of G. Assume for now $|S^* \cap V(G_t)| \geq 4/\delta$.

$$|S| \leq |S_t'| + |S'|$$
$$\leq |S_t| + 2(\ell + 1) + c \cdot (1 + \varepsilon)\mathrm{OPT}_{\mathrm{CVC}}(G')$$

Using $|S_t| \geq \frac{10\ell}{\delta}$

$$\leq |S_t| + \frac{\delta}{2}|S_t| + c \cdot (1 + \varepsilon)\mathrm{OPT}_{\mathrm{CVC}}(G')$$
$$\leq c \cdot (1 + \delta)(1 + \delta/2)\mathrm{OPT}_{\mathrm{CVC}}(G_t) + c \cdot (1 + \varepsilon)|(S^* \cap V(G')) \cup \{z_t\}|$$
$$\leq c \cdot (1 + \delta)(1 + \delta/2)|(S^* \cap V(G_t)) \cup \{z_t\}| + c \cdot (1 + \varepsilon)|(S^* \cap V(G')) \cup \{z_t\}|$$
$$\leq c \cdot (1 + \delta)(1 + \delta/2)(|S^* \cap V(G_t)| + 1) + c \cdot (1 + \varepsilon)|(S^* \cap V(G')) \cup \{z_t\}|$$

By assuming $|S^* \cap V(G_t)| \geq 4/\delta$, and then using $\delta = \varepsilon/3$

$$\leq c \cdot (1 + \delta)(1 + \delta/2)(1 + \delta/4)(|S^* \cap V(G_t)|) + c \cdot (1 + \varepsilon)|(S^* \cap V(G')) \cup \{z_t\}|$$
$$\leq c \cdot (1 + \varepsilon)(|S^* \cap V(G_t)|) + c \cdot (1 + \varepsilon)|(S^* \cap V(G')) \cup \{z_t\}|$$

Observe that since G_t and G' are non-empty, S^* must contain a vertex from X_t

$$\leq c \cdot (1 + \varepsilon)|S^*| = c \cdot (1 + \varepsilon) \cdot \mathrm{OPT}_{\mathrm{CVC}}(G).$$

It remains to observe that $|S^* \cap V(G_t)| \geq 4/\delta$ is a reasonable assumption. Suppose not, then $\mathrm{OPT}_{\mathrm{CVC}}(G_t) \leq |S^* \cap V(G_t)| + 1 \leq 4/\delta + 1$. However, $S_t \geq \frac{10\ell}{\delta} \geq 2 \cdot \mathrm{OPT}_{\mathrm{CVC}}(G_t)$, meaning that S_t is not a 2-approximation in G_t, which is a contradiction. ◁

Having shown the correctness of the procedure, it remains to argue the size of this Turing kernel. Observe that the oracle is only used when applying Claim 18. As such, we may bound the size of the kernel by $\mathcal{O}((\frac{\ell^2}{\delta})^{\lceil \frac{1+\delta}{\delta} \rceil}) = \mathcal{O}((\frac{\ell^2}{\varepsilon})^{\lceil \frac{3+\varepsilon}{\varepsilon} \rceil})$, recall that $\delta = \frac{\varepsilon}{3}$. ◀

4 Meta result

In this section we will describe a wide range of graph problems for which approximate Turing kernels can be obtained. The problems we will consider satisfy certain additional constraints, such that the general strategy already described for the VERTEX COVER problem can be applied. Informally speaking, we need the following requirements. First of all, the problems should behave nicely with respect to taking the disjoint union of graphs. Secondly, we want to look at what happens for induced subgraphs. We will only consider problems whose value cannot increase when taking an induced subgraph. Furthermore, we restrict how much the optimal value can decrease when taking an induced subgraph. Finally, we require existence of a PSAKS and an approximation algorithm for the problem. We use the following definitions.

▶ **Definition 21.** *Let $\varphi\colon \mathbb{R}\times\mathbb{N} \to \mathbb{R}$ be a function. A φ-approximation algorithm* for a problem $\mathcal{P}$ *is a polynomial-time algorithm that, given an instance G with tree decomposition $\mathcal{T}$ of width ℓ, outputs a solution S such that (for minimization problems) $\mathcal{P}(G,S) \le \varphi(\mathrm{OPT}_{\mathcal{P}}(G), \ell)$, and (for maximization problems) $\varphi(\mathcal{P}(G,S), \ell) \ge \mathrm{OPT}_{\mathcal{P}}(G)$.*

▶ **Definition 22.** *Let $\mathcal{P}$ be an optimization problem whose input is a graph. We will say that it is* friendly *if it satisfies the following conditions.*

1. *For all graphs G, G_1, and G_2 such that G is the disjoint union of graphs G_1 and G_2, $\mathrm{OPT}_{\mathcal{P}}(G) = \mathrm{OPT}_{\mathcal{P}}(G_1) + \mathrm{OPT}_{\mathcal{P}}(G_2)$. In particular, if S_1 is a solution for G_1 and S_2 is a solution for G_2, then $S_1 \cup S_2$ is a solution for G and*

   $$\mathcal{P}(G, S_1 \cup S_2) = \mathcal{P}(G_1, S_1) + \mathcal{P}(G_2, S_2).$$

   In the other direction, given solution S in G it can efficiently be split into solutions S_1 in G_1 and S_2 in G_2 satisfying the above. For consistency, we require that the size of the optimal solution in the empty graph is zero.

2. *There exists a non-decreasing polynomial function f such that for all graphs G, for all $X \subseteq V(G)$:*

   $$\mathrm{OPT}_{\mathcal{P}}(G) \le \mathrm{OPT}_{\mathcal{P}}(G-X) + f(|X|), \text{ and } \mathrm{OPT}_{\mathcal{P}}(G-X) \le \mathrm{OPT}_{\mathcal{P}}(G).$$

   In particular, for minimization problems there is a polynomial-time algorithm $\mathcal{A}$ that, given a solution S' in $G - X$, outputs a solution S for G such that $\mathcal{P}(G,S) \le \mathcal{P}(G - X, S') + f(|X|)$. For maximization problems we require that any solution S for $G - X$ is also a solution for G and $\mathcal{P}(G,S) = \mathcal{P}(G - X, S)$.

3. *$\mathcal{P}^{\perp}$ parameterized by $k + \ell$, where k is the solution value and ℓ is the treewidth, has a $(1 + \delta)$-approximate kernel for all $\delta > 0$, that has $h(\delta, k + \ell)$ vertices for some function h that is polynomial in its second parameter.*

4. *$\mathcal{P}$ has a φ-approximation algorithm for some polynomial function φ such that $\alpha \cdot \varphi(k,\ell) < \varphi(\alpha \cdot k, \ell)$ for all $\alpha > 1$, and φ is non-decreasing in its first parameter.*

Observe that many well-known vertex subset problems fit in this framework. As an example, let us verify them for the Vertex Cover problem. The first point is immediate. For the second point, let $\mathcal{A}(G, X, S)$ output $S' := S \cup X$. Verify that indeed this satisfies the conditions with $f(|X|) = |X|$. The third point follows with some extra work from the fact that Vertex Cover has a kernel with $2k$ vertices, this kernel can then be shown to be 1-approximate. For the last point, it is well-known that Vertex Cover has a 2-approximation algorithm.

▶ **Lemma 23 (★).** *Let $\mathcal{P}$ be a friendly graph optimization problem. There is a polynomial-time algorithm $\mathcal{B}$ with access to a c-approximate oracle. It takes as input a graph G with nice tree decomposition $\mathcal{T}$ of width ℓ and a number $0 < \delta \le 1$, and outputs either*

- *a node t such that $\mathrm{OPT}_{\mathcal{P}}(G[V_t \setminus X_t]) \ge \frac{f(\ell+1)}{\delta}$ together with a $(c \cdot (1 + \delta))$-approximate solution S_t to $\mathcal{P}$ in $G[V_t \setminus X_t]$, or*
- *a $c \cdot (1 + \delta)$-approximate solution for G,*

using calls to the oracle on graphs with at most $h(\delta, \varphi(k, \ell) + \ell)$ vertices, where $k = \frac{2f(\ell+1)}{\delta} + f(1)$.

We will prove the result separately for maximization and minimization problems (see [24, Lemma 25] for the minimization case).

Proof of Lemma 23: Maximization problems. Let r be the root of $\mathcal{T}$, and observe that $G = G[V_r \setminus X_r]$ since $X_r = \emptyset$. Let $k := \frac{2f(\ell+1)}{\delta} + f(1)$. Compute a φ-approximate solution $\tilde{S}$ in G. We do a case distinction on the value of this solution.

If $\mathcal{P}(G, \tilde{S}) \leq k$, then apply the PSAKS with approximation ratio $1+\delta$ to $(G, \varphi(k, \ell)+\ell)$ and obtain instance (G', k') with at most $h(\delta, \varphi(k, \ell)+\ell)$ vertices. Obtain solution S' by applying the c-approximate oracle on G'. Apply the solution lifting algorithm to S' to obtain a solution S for G. We start by showing that S is the desired approximate solution. Clearly, $\mathcal{P}(G', S') \geq \frac{1}{c} \cdot \mathrm{OPT}_{\mathcal{P}}(G')$ by correctness of the oracle. If $\mathcal{P}(G', S') > k'$, then $\mathcal{P}^{\perp}(G', k', S') = k'+1$ and thus $\mathcal{P}^{\perp}(G', k', S') \geq \mathrm{OPT}_{\mathcal{P}^{\perp}}(G', k')$. Otherwise, we have $\mathcal{P}^{\perp}(G', k', S') = \mathcal{P}(G', S') \geq \frac{1}{c} \cdot \mathrm{OPT}_{\mathcal{P}}(G') \geq \frac{1}{c} \cdot \mathrm{OPT}_{\mathcal{P}^{\perp}}(G', k')$. From the properties of the solution lifting algorithm, it now follows that $\mathcal{P}^{\perp}(G, \varphi(k, \ell) + \ell, S) \geq \frac{1}{c(1+\delta)} \mathrm{OPT}_{\mathcal{P}^{\perp}}(G, \varphi(k, \ell) + \ell)$. Observe that since $\mathcal{P}(G, \tilde{S}) \leq k$ and φ non-decreasing in its first parameter, we get that $\mathrm{OPT}_{\mathcal{P}}(G) \leq \varphi(\mathcal{P}(G, \tilde{S}), \ell) \leq \varphi(k, \ell)$ and thereby $\mathrm{OPT}_{\mathcal{P}}(G) = \mathrm{OPT}_{\mathcal{P}^{\perp}}(G, \varphi(k, \ell) + \ell)$. It follows that $\mathcal{P}(G, S) \geq \mathcal{P}^{\perp}(G, \varphi(k, \ell) + \ell, S) \geq \frac{1}{c(1+\delta)} \mathrm{OPT}_{\mathcal{P}^{\perp}}(G, \varphi(k, \ell) + \ell) = \frac{1}{c(\delta+1)} \mathrm{OPT}_{\mathcal{P}}(G)$.

Suppose $\mathcal{P}(G, \tilde{S}) > k$. For every node $t \in T$, compute a φ-approximate solution $\tilde{S}_t$ for graph $G[V_t \setminus X_t]$. We start by showing how to find a node $t \in V(T)$ such that both $\mathcal{P}(G[V_t \setminus X_t], \tilde{S}_t) \leq k$, and $\mathrm{OPT}_{\mathcal{P}}(G[V_t \setminus X_t]) \geq \frac{f(\ell+1)}{\delta}$. Start by observing that for the leaf vertices, it holds that $\mathcal{P}(G[V_t \setminus X_t], \tilde{S}_t) = 0 \leq k$. On the other hand, for the root, we found that $\mathcal{P}(G[V_r \setminus X_r], \tilde{S}_r) = \mathcal{P}(G, \tilde{S}) > k$. As such, we can find a node p such that $\mathcal{P}(G[V_p \setminus X_p], \tilde{S}_p) > k$, while for all of its children t it holds that $\mathcal{P}(G[V_t \setminus X_t], \tilde{S}_t) \leq k$. We show that one of the children of p has the desired properties. The result that $\mathcal{P}(G[V_t \setminus X_t], \tilde{S}_t) \leq k$ for all children of p is immediate. On the other hand, observe that $\mathrm{OPT}_{\mathcal{P}}(G[V_p \setminus X_p]) \geq \mathcal{P}(G[V_p \setminus X_p], \tilde{S}_p) \geq k \geq \frac{2f(\ell+1)}{\delta}$, by assumption. We do a case distinction on the type of node that p is in the nice tree decomposition.

- p is an introduce or forget node. In this case, p has exactly one child t and $V_t \setminus X_t = V_p \setminus X_p$, or $V_t \setminus X_t = (V_p \setminus X_p) \setminus \{v\}$ for some $v \in V(G)$. Since $\mathcal{P}$ is friendly, we get that $\mathrm{OPT}_{\mathcal{P}}(G[V_t \setminus X_t]) \geq \mathrm{OPT}_{\mathcal{P}}(G[V_p \setminus X_p]) - f(1) \geq \frac{f(\ell+1)}{\delta}$.

- p is a join node. In this case, p has exactly two children t_1 and t_2 and $G[V_p \setminus X_p]$ is the disjoint union of $G[V_{t_1} \setminus X_{t_1}]$ and $G[V_{t_2} \setminus X_{t_2}]$. Obtain S_1 and S_2 such that $\tilde{S}_p = S_1 \cup S_2$ and S_1 is a solution in $G[V_{t_1} \setminus X_{t_1}]$, S_2 in $G[V_{t_2} \setminus X_{t_2}]$, and $\mathcal{P}(G[V_p \setminus X_p], \tilde{S}_p) = \mathcal{P}(G[V_{t_1} \setminus X_{t_1}], S_1) + \mathcal{P}(G[V_{t_2} \setminus X_{t_2}], S_2)$. This can be done since $\mathcal{P}$ is friendly.
  Therefore, there is $i \in [2]$ such that $\mathrm{OPT}_{\mathcal{P}}(G[V_{t_i} \setminus X_{t_i}]) \geq \mathcal{P}(G[V_{t_i} \setminus X_{t_i}], S_i) \geq \mathcal{P}(G[V_p \setminus X_p], \tilde{S}_p)/2 \geq \frac{f(\ell+1)}{\delta}$.

So, we have obtained a node t such that $\mathcal{P}(G[V_t \setminus X_t], \tilde{S}_t) \leq k$, and $\mathrm{OPT}_{\mathcal{P}}(G[V_t \setminus X_t]) \geq \frac{f(\ell+1)}{\delta}$. We now show how to obtain S_t. Apply the PSAKS with ratio $1 + \delta$ to $(G[V_t \setminus X_t], \varphi(k, \ell) + \ell)$ and obtain instance (G', k'). Apply the c-approximate oracle on G' to obtain a solution S''. Apply the solution lifting algorithm to S'' to obtain solution S_t in $G[V_t \setminus X_t]$. With similar arguments as before, S_t is a $c(1 + \delta)$-approximate solution in $G[V_t \setminus X_t]$. Output t and S_t. ◀

The next theorem gives a polynomial-size $(1+\varepsilon)$-approximate Turing kernel with parameter treewidth for any friendly optimization problem $\mathcal{P}$. The Turing kernel follows the same ideas as the Turing kernels presented earlier in this paper, using Lemma 23 to find a node in the tree decomposition where we can split the graph.

▶ **Theorem 24 (★).** *Let $\mathcal{P}$ be a friendly optimization problem on graphs. Then $\mathcal{P}$ parameterized by treewidth has a $(1+\varepsilon)$-approximate Turing kernel with $h\left(\frac{\varepsilon}{3}, \varphi\left(\frac{6f(\ell+1)}{\varepsilon} + f(1), \ell\right) + \ell\right)$ vertices, for all $0 < \varepsilon \leq 1$.*

While the description of the Turing kernel is mostly the same for maximization and minimization problems, the correctness proof will differ quite significantly. Therefore, these cases will be proven separately, the proof for minimization problems can be found in the full version of the paper.

Proof of Theorem 24: Maximization problems. Let $\mathcal{P}$ be a friendly maximization problem. We show that Algorithm 3 is the desired approximate Turing kernel, where we let $\mathcal{A}(G, X_t, S' \cup S_t)$ return $S' \cup S_t$.

■ **Algorithm 3** An approximate Turing kernel for friendly optimization problems $\mathcal{P}$.

1: **procedure** APPROXP$(G, \mathcal{T}, \varepsilon)$
2:     Turn $\mathcal{T}$ into a nice tree decomposition
3:     Apply Lemma 23 for $\delta := \varepsilon/3$
4:     **if** this outputs an approximate solution S for G **then**
5:         **return** S
6:     **else** // *We obtained $t \in V(T)$, $c(1 + \delta)$-approximate solution S_t for $\mathcal{P}$ in $G[V_t \setminus X_t]$*
           such that $\text{OPT}_{\mathcal{P}}(G[V_t \setminus X_t]) \geq \frac{f(\ell+1)}{\delta}$
7:         Let $G' := G - V_t$.
8:         Obtain $\mathcal{T}'$ from $\mathcal{T}$ by removing the subtree rooted at t and all vertices in X_t
9:         Let $S' := \text{APPROXP}(G', \mathcal{T}', \varepsilon)$
10:         **return** $S := \mathcal{A}(G, X_t, S' \cup S_t)$
11:     **end if**
12: **end procedure**

It is easy to see that since $\mathcal{P}$ is friendly, the algorithm indeed returns a correct solution for $\mathcal{P}$ in G, it remains to prove the size bound.

$$
\begin{aligned}
\text{OPT}_{\mathcal{P}}(G) &\leq \text{OPT}_{\mathcal{P}}(G - X_t) + f(\ell + 1) \\
&= \text{OPT}_{\mathcal{P}}(G - V_t) + \text{OPT}_{\mathcal{P}}(G[V_t \setminus X_t]) + f(\ell + 1) \\
&\leq \text{OPT}_{\mathcal{P}}(G - V_t) + (1 + \delta) \cdot \text{OPT}_{\mathcal{P}}(G[V_t \setminus X_t]) \\
&\leq c \cdot (1 + \varepsilon) \cdot \mathcal{P}(G - V_t, S') + c \cdot (1 + \delta)^2 \cdot \mathcal{P}(G[V_t \setminus X_t], S_t) \\
&\leq c \cdot (1 + \varepsilon) \cdot (\mathcal{P}(G - V_t, S') + \mathcal{P}(G[V_t \setminus X_t], S_t)) \\
&= c \cdot (1 + \varepsilon) \cdot (\mathcal{P}(G - X_t, S' \cup S_t)) = c \cdot (1 + \varepsilon) \cdot (\mathcal{P}(G, S' \cup S_t)). \qquad \blacktriangleleft
\end{aligned}
$$

4.1   Consequences

We show that a number of considered problems are friendly in the next lemma.

▶ **Lemma 25.** *The following problems are friendly (with respect to the following bounds).*

▬ INDEPENDENT SET *with* $f(x) = x$, $h(\delta, m) = (m + 1)^2$, $\varphi(s, \ell) = (\ell + 1) \cdot s$.

▬ VERTEX-DISJOINT H-PACKING *for connected graphs H, with $|V(H)|$ constant, with* $f(x) = x$, $h(\delta, k) = \mathcal{O}(k^{|V(H)|-1})$, $\varphi(s, \ell) = |V(H)| \cdot s$.

▬ VERTEX COVER *with* $f(x) = x$, $h(\delta, k) = 2k$, $\varphi(s, \ell) = 2s$.

▬ CLIQUE COVER *with* $f(x) = x$, $h(\delta, m) = m(m + 1)$, $\varphi(s, \ell) = (\ell + 1) \cdot s$

▬ FEEDBACK VERTEX SET *with* $f(x) = x$, $h(\delta, k) = 4k^2$, $\varphi(s, \ell) = 2s$.

▬ EDGE DOMINATING SET *with* $f(x) = x$, $h(\delta, k) = 4k^2 + 4k$, $\varphi(s, \ell) = 2s$.

Proof.

Independent Set. Clearly, if G is the disjoint union of two graphs G_1 and G_2, then the union of an independent set in G_1 and an independent set in G_2 forms an independent set in G. Conversely, restricting an independent set in G to $V(G_1)$ (respectively $V(G_2)$) results in an independent set in G_1 (respectively, G_2). Furthermore, if X is a subset of G it is easy to verify that $\mathrm{OPT}_{\mathrm{IS}}(G) \leq \mathrm{OPT}_{\mathrm{IS}}(G - X) + |X|$ and that $\mathrm{OPT}_{\mathrm{IS}}(G - X) \leq \mathrm{OPT}_{\mathrm{IS}}(G)$ as any independent set in $G - X$ is an independent set in G. The PSAKS parameterized by $m := k + \ell$ is as follows. It is known that any graph of treewidth ℓ has an independent set of size at least $|V(G)|/(\ell + 1)$. This can be seen from the fact that such graphs are ℓ-degenerate, meaning that there is an order of the vertices $v_1, \ldots, v_n$ such that v_i has degree at most ℓ in $G[v_1, \ldots, v_i]$. As such, an independent set of size $|V(G)|/(\ell + 1)$ can be greedily constructed.

Thus, if $|V(G)| > (m + 1)^2$, we simply let G' be the graph consisting of an independent set of size $m + 1$. The solution lifting algorithm can then simply find a size-$(m + 1)$ independent set and output it. This is always an optimal solution for $\mathcal{P}^\perp$, since it does not distinguish between solutions of size larger than m. Otherwise, we obtain that $|V(G)| \leq (m + 1)^2$ and the PSAKS will not modify G. In both cases, we output a graph on at most $(m + 1)^2$ vertices.

It remains to show that there is an approximation algorithm, the idea is equivalent to the PSAKS. Return an independent set in G of size at least $|V(G)|/(\ell + 1)$. Then indeed $\varphi(|V(G)|/(\ell + 1), \ell) = |V(G)| \geq \mathrm{OPT}_{\mathrm{IS}}(G)$.

Vertex-Disjoint H-Packing. Requirements 1 and 2 are easily verified for $f(|X|) = |X|$, as any vertex in X could be contained in at most one graph in any copy of H.

A simple approximation algorithm for VERTEX-DISJOINT H-PACKING is to simply return any maximal H-packing S. We show that $|S| \geq \frac{1}{|V(H)|}\mathrm{OPT}_{\mathcal{P}}(G)$, such that this is an φ-approximation algorithm with $\varphi(s, \ell) = |V(H)| \cdot s$. Suppose there is an optimal solution S^* with $|S^*| > |V(H)| \cdot |S|$. Since the copies of H in S are vertex-disjoint, S uses exactly $|V(H)| \cdot |S|$ vertices. Since S^* contains more than $|V(H)| \cdot |S|$ elements, it follows that there is $s \in S^*$ that uses no vertices used by S, contradicting that S is maximal.

The existence of a PSAKS is shown in [24, Lemma 31, Appendix A].

Vertex Cover. Requirements 1 and 2 are easily verified for vertex cover, let algorithm $\mathcal{A}$ simply output the union of the given solution with set X. As (implicitly) observed in the proof of Lemma 8, VERTEX COVER has a 1-approximate kernel of size $2k$. Furthermore, it is well-known to be 2-approximable.

Clique Cover. Requirement 1 is easy to verify. We show Requirement 2. Let $X \subseteq V(G)$. Let S be a clique cover of G, it is easy to see that $\{s \setminus X \mid s \in S\}$ is a clique cover of $G - X$, of size at most $|S|$. Therefore, $\mathrm{OPT}_{\mathcal{P}}(G) \geq \mathrm{OPT}_{\mathcal{P}}(G - X)$. Furthermore, let algorithm $\mathcal{A}$ when given G, clique cover S of $G - X$ and X output the clique cover $S \cup \{\{x\} \mid x \in X\}$. Then this is a clique cover of G and it has size at most $|S| + |X| \leq |S| + f(|X|)$.

To show Requirement 3, we obtain a 1-approximate kernel for CLIQUE COVER in a somewhat similar way as for INDEPENDENT SET. Observe that any n-vertex graph with treewidth ℓ has a minimum clique cover of size at least $\frac{n}{\ell+1}$. So, given G and parameter $m := k + \ell$, if $n > m(m + 1) \geq k \cdot (\ell + 1)$, we know for sure that G does not have a minimum clique cover of size k. The reduction algorithm reduces G to an independent set of size $m + 1$. The solution lifting algorithm (irrespective of the solution given for G') outputs $V(G)$. Otherwise, if $n \leq m(m + 1)$ we simply let G be the output of the reduction algorithm. Since the graph does not change, the solution lifting algorithm simply outputs the solution it is given. In both cases, the reduced instance has size at most $m(m + 1)$.

It remains to verify that there is a φ-approximation algorithm for CLIQUE COVER. Given a graph G of treewidth ℓ, we simply output $\{\{v\} \mid v \in V(G)\}$. Clearly, this is a valid clique cover of G of size $|V(G)|$. Observe that since G has treewidth ℓ, G contains no cliques of size larger than $\ell + 1$, thus any clique in the optimal clique cover of G covers at most $\ell + 1$ vertices. As such, the optimal solution contains at least $\frac{|V(G)|}{\ell+1}$ cliques, and thus $|S| \leq (\ell + 1)\mathrm{OPT}_{\mathcal{P}}(G)$.

Feedback Vertex Set. Requirements 1 and 2 are straightforward to verify. The problem has a 1-approximate kernel with $4k^2$ vertices and therefore a PSAKS by [24, Lemma 29, Appendix A], showing Requirement 3. It is also known that the FEEDBACK VERTEX SET problem has a 2-approximation algorithm [2], showing Requirement 4.

Edge Dominating Set. Requirement 1 is again straightforward. For the second requirement, let G be a graph and let $X \subseteq V(G)$. We start by showing that $\mathrm{OPT}_{\mathcal{P}}(G) \geq \mathrm{OPT}_{\mathcal{P}}(G - X)$. Let S be an edge-dominating set in G. We obtain an edge-dominating set S' for $G - X$ as follows. Initialize S' as the set of edges with both endpoints in $V(G) \setminus X$, so $S' := \{e \in S \mid e \cap X = \emptyset\}$. For every edge $\{x, v\} \in S$ with $x \in X, v \notin X$, choose one arbitrary edge $\{u, v\} \in E(G - X)$ and add $\{u, v\}$ to S'. If no such edge exists, do nothing. Clearly, $|S'| \leq |S|$. Furthermore, we show that S' is indeed an edge dominating set. Suppose for contradiction that $e = \{u, v\}$ is not dominated in $G - X$ by S'. Let $\{w, v\} \in S$ be the edge dominating $\{u, v\}$ in G. Then, since $\{w, v\} \notin S'$, we have $w \in X$. But then some edge with endpoint v was added to S', a contradiction.

We continue by showing that $\mathrm{OPT}_{\mathcal{P}}(G) \leq \mathrm{OPT}_{\mathcal{P}}(G - X) + |X|$ and that algorithm $\mathcal{A}$ exists. Let S be a solution for $G - X$, then algorithm $\mathcal{A}$ will output S together with one edge $\{x, v\} \in E(G)$ for all $x \in X$. In the case that $x \in X$ is isolated in G, no edge is added for this vertex. By this definition, the output has size at most $|S| + |X|$. Furthermore, any edge with vertices in $V(G - X)$ is dominated by S. Any edge with at least one endpoint in X is dominated by the additional edges.

EDGE DOMINATING SET has a kernel that outputs a graph G' of size at most $4k^2 + 4k$ such that G' is an induced subgraph of G and any size-k edge dominating set in G' is also an edge dominating set in G [21]. We can see that this is a 1-approximate kernel. Let the solution lifting algorithm simply output the solution for G' as a solution for G. Since any solution of size at most k in G' is a solution in G, and obviously any solution in G is a solution for G', it is clear that $\mathrm{OPT}_{\mathcal{P}\perp}(G', k) = \mathrm{OPT}_{\mathcal{P}\perp}(G, k)$. As such, the approximation ratio is preserved by the solution lifting algorithm.

It is known that even the weighted version of EDGE DOMINATING SET can be 2-approximated [20], such that the problem has a φ-approximation for $\varphi(s, \ell) = 2s$.   ◄

As an immediate consequence of Lemma 25 and Theorem 24, we obtain approximate Turing kernels for a large number of graph problems. These results are summarized in the corollary below, the size bounds are obtained by substituting the relevant bounds given by Lemma 25 into Theorem 24.

▶ **Corollary 26.** *The following problems have a polynomial $(1+\varepsilon)$-approximate Turing kernel for all $0 < \varepsilon \leq 1$, of the given size (in number of vertices), when parameterized by treewidth ℓ.*

- INDEPENDENT SET, *of size* $\mathcal{O}(\frac{\ell^4}{\varepsilon^2})$.
- VERTEX-DISJOINT H-PACKING *for connected graphs H, of size* $\mathcal{O}((\frac{\ell}{\varepsilon})^{|V(H)|-1})$.
- VERTEX COVER *of size* $\mathcal{O}(\frac{\ell}{\varepsilon})$.
- CLIQUE COVER *of size* $\mathcal{O}(\frac{\ell^4}{\varepsilon^2})$.
- FEEDBACK VERTEX SET *of size* $\mathcal{O}((\frac{\ell}{\varepsilon})^2)$.
- EDGE DOMINATING SET *of size* $\mathcal{O}((\frac{\ell}{\varepsilon})^2)$.

We observe that the bounds for Independent Set and Clique Cover can be improved to $\mathcal{O}(\frac{\ell^2}{\varepsilon})$ by a more careful analysis. Instead of using that the problem is friendly and applying Lemma 23, one may simply find t such that the number of vertices in $G[V_t \setminus X_t]$ is between $\frac{(\ell+1)^2}{\delta}$ and $\frac{10(\ell+1)^2}{\delta}$, and use that an optimal solution has size at least $|V(G)|/(\ell+1)$ for graphs of treewidth ℓ. There is no need to apply a kernelization in this case.

5 Conclusion

In this paper we have provided approximate Turing kernels for various graph problems when parameterized by treewidth. Furthermore, we give a general result that can be used to obtain approximate Turing kernels for all friendly graph problems parameterized by treewidth.

While the notion of being friendly captures many known graph problems, some interesting problems do not fit this definition. In particular, it is not clear whether the Dominating Set problem has a polynomial-size constant-factor approximate Turing kernel when parameterized by treewidth. We leave this as an open problem.

References

1 Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Interval vertex deletion admits a polynomial kernel. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1711–1730. SIAM, 2019. `doi:10.1137/1.9781611975482.103`.

2 Ann Becker and Dan Geiger. Optimization of pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188, 1996. `doi:10.1016/0004-3702(95)00004-6`.

3 Daniel Binkele-Raible, Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Trans. Algorithms*, 8(4):38:1–38:19, 2012. `doi:10.1145/2344422.2344428`.

4 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. `doi:10.1016/j.jcss.2009.04.001`.

5 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (meta) kernelization. *J. ACM*, 63(5):44:1–44:69, 2016. `doi:10.1145/2973749`.

6 Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. `doi:10.1016/j.tcs.2011.04.039`.

7 Yixin Cao, Ashutosh Rai, R. B. Sandeep, and Junjie Ye. A polynomial kernel for diamond-free editing. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, volume 112 of *LIPIcs*, pages 10:1–10:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.ESA.2018.10`.

8 Steven Chaplick, Fedor V. Fomin, Petr A. Golovach, Dusan Knop, and Peter Zeman. Kernelization of graph hamiltonicity: Proper h-graphs. In Zachary Friggstad, Jörg-Rüdiger Sack, and Mohammad R. Salavatipour, editors, *Algorithms and Data Structures - 16th International Symposium, WADS 2019, Edmonton, AB, Canada, August 5-7, 2019, Proceedings*, volume 11646 of *Lecture Notes in Computer Science*, pages 296–310. Springer, 2019. `doi:10.1007/978-3-030-24766-9_22`.

9 Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. In Peter Widmayer, Gabriele Neyer, and Stephan Eidenbenz, editors, *Graph-Theoretic Concepts in Computer Science*, pages 313–324, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

10   Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.

11   Marek Cygan, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Magnus Wahlström. Clique cover and graph separation: New incompressibility results. *ACM Trans. Comput. Theory*, 6(2):6:1–6:19, 2014. `doi:10.1145/2594439`.

12   Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and IDs. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014. `doi:10.1145/2650261`.

13   Eduard Eiben, Danny Hermelin, and M. S. Ramanujan. Lossy kernels for hitting subgraphs. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 67:1–67:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. `doi:10.4230/LIPIcs.MFCS.2017.67`.

14   Eduard Eiben, Mithilesh Kumar, Amer E. Mouawad, Fahad Panolan, and Sebastian Siebertz. Lossy kernels for connected dominating set on sparse graphs. *SIAM J. Discrete Math.*, 33(3):1743–1771, 2019. `doi:10.1137/18M1172508`.

15   Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, 38(2):629–657, 2008. `doi:10.1137/05064299X`.

16   Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 470–479. IEEE Computer Society, 2012. `doi:10.1109/FOCS.2012.62`.

17   Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. `doi:10.1017/9781107415157`.

18   Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct pcps for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. `doi:10.1016/j.jcss.2010.06.007`.

19   Pierre Fraigniaud and Nicolas Nisse. Connected treewidth and connected graph searching. In *LATIN 2006: Theoretical Informatics, 7th Latin American Symposium, Valdivia, Chile, March 20-24, 2006, Proceedings*, pages 479–490, 2006. `doi:10.1007/11682462_45`.

20   Toshihiro Fujito and Hiroshi Nagamochi. A 2-approximation algorithm for the minimum weight edge dominating set problem. *Discrete Applied Mathematics*, 118(3):199–207, 2002. `doi:10.1016/S0166-218X(00)00383-8`.

21   Torben Hagerup. Kernels for edge dominating set: Simpler or smaller. In *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings*, pages 491–502, 2012. `doi:10.1007/978-3-642-32589-2_44`.

22   Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. `doi:10.1007/s00453-014-9910-8`.

23   Eva-Maria C. Hols and Stefan Kratsch. On kernelization for edge dominating set under structural parameters. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 36:1–36:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.STACS.2019.36`.

24   Eva-Maria C. Hols, Stefan Kratsch, and Astrid Pieterse. Approximate turing kernelization for problems parameterized by treewidth. *CoRR*, abs/2004.12683, 2020. `arXiv:2004.12683v1`.

25   Bart M. P. Jansen. Turing kernelization for finding long paths and cycles in restricted graph classes. *J. Comput. Syst. Sci.*, 85:18–37, 2017. `doi:10.1016/j.jcss.2016.10.008`.

26   Bart M. P. Jansen and Dániel Marx. Characterizing the easy-to-find subgraphs from the viewpoint of polynomial-time algorithms, kernels, and turing kernels. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 616–629. SIAM, 2015. `doi: 10.1137/1.9781611973730.42`.

27   Bart M. P. Jansen and Astrid Pieterse. Polynomial kernels for hitting forbidden minors under structural parameterizations. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, volume 112 of *LIPIcs*, pages 48:1–48:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.ESA.2018.48`.

28   Bart M. P. Jansen, Marcin Pilipczuk, and Erik Jan van Leeuwen. A deterministic polynomial kernel for odd cycle transversal and vertex multiway cut in planar graphs. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 39:1–39:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.STACS.2019.39`.

29   Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 450–459. IEEE Computer Society, 2012. `doi:10.1109/FOCS.2012.46`.

30   Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 224–237, 2017. `doi:10.1145/3055399. 3055456`.

31   Rolf Niedermeier and Christophe Paul, editors. *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: `http://www. dagstuhl.de/dagpub/978-3-95977-100-9`.

32   M. S. Ramanujan. An approximate kernel for connected feedback vertex set. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 77:1–77:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi: 10.4230/LIPIcs.ESA.2019.77`.

33   Carla D. Savage. Depth-first search and the vertex cover problem. *Inf. Process. Lett.*, 14(5):233–237, 1982. `doi:10.1016/0020-0190(82)90022-9`.

34   Stéphan Thomassé, Nicolas Trotignon, and Kristina Vuskovic. A polynomial turing-kernel for weighted independent set in bull-free graphs. *Algorithmica*, 77(3):619–641, 2017. `doi: 10.1007/s00453-015-0083-x`.

35   Jouke Witteveen, Ralph Bottesch, and Leen Torenvliet. A hierarchy of polynomial kernels. In Barbara Catania, Rastislav Královic, Jerzy R. Nawrocki, and Giovanni Pighizzini, editors, *SOFSEM 2019: Theory and Practice of Computer Science - 45th International Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 27-30, 2019, Proceedings*, volume 11376 of *Lecture Notes in Computer Science*, pages 504–518. Springer, 2019. `doi:10.1007/978-3-030-10801-4_39`.