Lossy Kernels for Hitting Subgraphs[*]

Eduard Eiben[1], Danny Hermelin[2], and M. S. Ramanujan[3]

1 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
`eiben@ac.tuwien.ac.at`

2 **Industrial Engineering and Management, Ben Gurion University, Be'er Scheva, Israel**
`hermelin@bgu.ac.il`

3 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
`ramanujan@ac.tuwien.ac.at`

Abstract

In this paper, we study the Connected $\mathcal{H}$-hitting Set and Dominating Set problems from the perspective of *approximate* kernelization, a framework recently introduced by Lokshtanov et al. [STOC 2017]. For the Connected $\mathcal{H}$-hitting set problem, we obtain an α-approximate kernel for every $\alpha > 1$ and complement it with a lower bound for the natural weighted version. We then perform a refined analysis of the tradeoff between the approximation factor and kernel size for the Dominating Set problem on d-degenerate graphs, and provide an interpolation of approximate kernels between the known d^2-approximate kernel of constant size and 1-approximate kernel of size $k^{\mathcal{O}(d^2)}$.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity, G.2.1 Combinatorics

Keywords and phrases parameterized algorithms, lossy kernelization, graph theory

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.67

1 Introduction

Polynomial time preprocessing is one of the widely used methods to tackle NP-hardness in practice and the area of *kernelization* has been extremely successful in laying down a mathematical framework for the design and rigorous analysis of preprocessing algorithms for *decision problems*. We refer the reader to the survey articles by Kratsch [17] or Lokshtanov et al. [18] for recent developments, or the textbooks [6, 11] for an introduction to the field. The central notion in kernelization is that of a *kernel*, which is a preprocessing algorithm that runs in polynomial time and transforms a 'large' instance of a decision problem into a significantly smaller, but equivalent instance.

Unfortunately, the existing notion of kernels, having been built around decision problems, does not combine well with approximation algorithms and heuristics. In particular, in order for kernels to be useful, one is required to solve the preprocessed instance exactly. However, this may not always be possible and the existing theory of kernelization says nothing about being able to infer useful information from a good *approximate* solution for the preprocessed instance. Lokshtanov et al. [19] attempted to address this limitation by introducing the notion of *α-approximate kernels*. Informally speaking, an α-approximate kernel is a polynomial time algorithm that given an instance (I, k) outputs an instance (I', k') such that $|I'| + k' \leq g(k)$

[*] This work was partially supported by the Austrian Science Fund (FWF, projects P26696 and W1255-N23).

for some computable function g and any c-approximate solution to the instance (I', k') can be turned in polynomial time into a $(c \cdot \alpha)$-approximate solution to the original instance (I, k). The function g is ideally polynomially bounded, in which case we call this algorithm, an α-approximate polynomial kernel. We refer the reader to the section on Preliminaries for a formal definition of the terms involved.

In their work, Lokshtanov et al. considered several problems which are known to not admit a polynomial kernel and showed that they do have an α-approximate polynomial kernel for *every* fixed $\alpha > 1$. They also proposed a machinery for proving lower bounds and managed to rule out even α-approximate kernels for some basic problems such as Longest Path and Set Cover under standard complexity theoretic hypotheses.

One of the fundamental classes of problems known to exclude polynomial kernels is the class of 'subgraph hitting' problems *with a connectivity constraint*. It is well-known that placing connectivity constraints on certain subgraph hitting problems can have a dramatic effect on their amenability to preprocessing. A case in point is the classic Vertex Cover problem. This problem is known to admit a kernel with $\mathcal{O}(k)$ vertices [6]. However, the Connected Vertex Cover problem is among the earliest problems to be shown to exclude a polynomial kernel [10] and Lokshtanov et al. [19] showed that this problem admits an α-approximate polynomial kernel for every $\alpha > 1$. Their result motivates the need to obtain a finer understanding of the role played by connectivity constraints in relation to preprocessing for other subgraph hitting problems. Therefore, a systematic study of the approximate kernelization of subgraph hitting problems with connectivity constraints is a natural strategy towards achieving this goal.

Our Contributions. In this paper, we study the Connected $\mathcal{H}$-Hitting Set (Conn-$\mathcal{H}$-HS) problem where the input is a graph G and an integer k and the objective is to check whether there is a set S of at most k vertices such that $G[S]$ is connected and $G - S$ has no vertex-induced subgraph isomorphic to a graph in the fixed finite family of graphs, $\mathcal{H}$. It is easy to see that this problem generalizes Connected Vertex Cover (set $\mathcal{H} = \{K_2\}$) and hence it is unlikely to have a polynomial kernel. As a result, we consider the approximate kernelization complexity of the *optimization* version of this problem and provide two results; one positive and one negative. Our positive result generalizes the approximate kernel given by Lokshtanov et al. for the Connected Vertex Cover problem and shows that Conn-$\mathcal{H}$-HS also admits an α-approximate polynomial kernel for *every* constant $\alpha > 1$ and fixed $\mathcal{H}$.

► **Theorem 1.** *For every fixed $\epsilon > 0$, there is a $(1 + \epsilon)$-approximate polynomial kernel for* Connected $\mathcal{H}$-hitting set.

Our negative result shows that this ability to obtain approximate kernels vanishes in the presence of weights, even when the domain of the weight function is highly restricted. To be precise, we study the Weighted Conn-$\mathcal{H}$-HS problem where the input also includes a weight function on the vertices and the objective now is to minimize the total *weight* of a connected set of vertices hitting all vertex-induced subgraphs of G isomorphic to a graph in $\mathcal{H}$. We show that unless $\mathsf{NP} \subseteq \mathsf{coNP/Poly}$, this problem has no α-approximate polynomial kernel for *any* constant α even when the domain of the weight function is restricted to $\{0, 1\}$. The formal statement of this theorem requires certain terms which are as yet undefined and we refer the reader to Section 3.2 for the statement.

In the second part of the paper, we initiate the fine-grained analysis of the accuracy-size tradeoff encountered when designing approximate kernels for the Dominating Set problem on the class of d-degenerate graphs. The Dominating Set problem is one of the most fundamental problems in algorithmic graph theory. In the decision version of this problem, the input is a graph G, an integer k and the objective is to decide whether G has a set S of at most k vertices which contains at least one neighbor of every vertex in $V(G) \setminus S$. It is well-known that Dominating Set is W[2]-hard [6], implying that it cannot have *any* kernel under standard complexity theoretic hypotheses. This fact motivated the study of preprocessing for this basic problem on restricted graph classes, leading to a long and rich literature [1, 2, 3, 7, 8, 12, 15, 20].

One of the more general graph classes on which Dominating Set is known to admit a polynomial kernel, is the class of d-degenerate graphs, which contains several well-studied graph classes such as planar graphs, graphs of bounded treewidth, graphs of bounded arboricity, and graphs excluding a fixed (topological) minor. Alon and Gutner [2] initiated the study of the parameterized complexity of Dominating Set on d-degenerate graphs and Philip et al. [20] obtained a kernel of size $k^{\mathcal{O}(d^2)}$, which was the first polynomial kernel for this problem on d-degenerate graphs. In fact, it follows from their proofs that this kernel is in fact a 1-approximate kernel. At the other extreme, it follows from [16] that there is a d^2-approximate kernel of *constant size*. These two results motivate the natural question: *What is the precise tradeoff between accuracy and kernel size for* Dominating Set *on d-degenerate graphs?* We give a sequence of approximate kernels for this problem which lie 'between' the two extremes and provide an interesting interpolation of kernels.

▶ **Theorem 2.** Dominating Set *on d-degenerate graphs has a $\lceil \frac{d}{\rho} \rceil$-approximate kernel of size $k^{\mathcal{O}(d\rho)}$, for any fixed integer $\rho \in \{1, \dots, d\}$.*

All approximate kernels obtained thus far (including that in the first part of our paper) are focussed on problems which are known to not admit polynomial kernels. Our work on Dominating Set on d-degenerate graphs thus initiates and motivates the fine-grained study of approximate kernelization even for problems which have polynomial kernels. Therefore, we believe that our result opens up a new line of investigation in the topic of approximate kernelization.

2 Preliminaries

The notion of kernels is based on *parameterized problems* from the area of Parameterized Complexity [6, 11]. Inputs of a parameterized problem are of the form (I, k) where I is a bitstring encoding an instance and k is an integer called the *parameter*, and every input is either a yes instance or a no instance. A *kernel* is a polynomial time algorithm that given an instance (I, k) of a parameterized problem outputs an instance (I', k') of the same problem such that $|I'| + k' \leq g(k)$ for some computable function g and (I, k) is a yes instance if and only if (I', k') is a yes instance. We now recall the main definitions from [19] regarding *parameterized optimization problems and approximate kernels.*

▶ **Definition 3** ([19]). A parameterized optimization (minimization or maximization) problem Π is a computable function $\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$.

The *instances* of a parameterized optimization problem Π are pairs $(I, k) \in \Sigma^* \times \mathbb{N}$, and a *solution* to (I, k) is simply a string $s \in \Sigma^*$, such that $|s| \leq |I| + k$. The *value* of the solution s is $\Pi(I, k, s)$. Since the problems we deal with in this paper are all minimization problems, we state some of the definitions only in terms of minimization problems when the definition for maximization problems is analogous.

▶ **Definition 4** ([19]). For a parameterized minimization problem Π, the *optimum value* of an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ is $OPT_\Pi(I, k) = \min_{\substack{s \in \Sigma^* \\ |s| \le |I|+k}} \Pi(I, k, s)$.

▶ **Definition 5** ([19]). Let $\alpha \ge 1$ be a real number and Π be a parameterized minimization problem. An *α-approximate polynomial time preprocessing algorithm $\mathcal{A}$* for Π is a pair of polynomial time algorithms. The first one is called the *reduction algorithm*, and computes a map $\mathcal{R}_\mathcal{A} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$. Given as input an instance (I, k) of Π the reduction algorithm outputs another instance $(I', k') = \mathcal{R}_\mathcal{A}(I, k)$.

The second algorithm is called the *solution lifting algorithm.* This algorithm takes as input an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ of Π, the output instance (I', k') of the reduction algorithm, and a solution s' to the instance (I', k'). The solution lifting algorithm works in time polynomial in $|I|$,k,$|I'|$,k' and s', and outputs a solution s to (I, k) such that the following holds.

$$\frac{\Pi(I, k, s)}{OPT(I, k)} \le \alpha \cdot \frac{\Pi(I', k', s')}{OPT(I', k')}.$$

The *size* of a polynomial time preprocessing algorithm $\mathcal{A}$ is a function $\text{size}_\mathcal{A} : \mathbb{N} \to \mathbb{N}$ defined as follows.

$$\text{size}_\mathcal{A}(k) = \sup\{|I'| + k' : (I', k') = \mathcal{R}_\mathcal{A}(I, k), I \in \Sigma^*\}.$$

▶ **Definition 6** ([19]). An *α-approximate kernelization* (or *α-approximate kernel*) for a parameterized optimization problem Π, and real $\alpha \ge 1$, is an α-approximate polynomial time preprocessing algorithm $\mathcal{A}$ for Π such that $\text{size}_\mathcal{A}$ is upper bounded by a computable function $g : \mathbb{N} \to \mathbb{N}$. We say that $\mathcal{A}$ is an α-approximate polynomial kernelization if g is a polynomial function.

▶ **Definition 7** ([19]). Let $\alpha \ge 1$ be a real number, and Π be a parameterized minimization problem. An α-approximate polynomial time preprocessing algorithm for Π is said to be *strict* if, for every instance (I, k), reduced instance $(I', k') = \mathcal{R}_\mathcal{A}(I, k)$ and solution s' to (I', k'), the solution s to (I, k) output by the solution lifting algorithm when given s' as input satisfies the following.

$$\frac{\Pi(I, k, s)}{OPT(I, k)} \le \max\left\{\frac{\Pi(I', k', s')}{OPT(I', k')}, \alpha\right\}$$

The notion of *strictness* in the above direction allows one to 'chain' multiple α-approximate preprocessing algorithms to obtain a single α-approximate preprocessing algorithm.

▶ **Definition 8.** A *reduction rule* is simply the reduction algorithm of a polynomial time preprocessing algorithm. The reduction rule *applies* if the output instance of the reduction algorithm is not the same as the input instance.

▶ **Definition 9** ([19]). A reduction rule is *α-safe for* Π if it is the reduction algorithm of a strict α-approximate polynomial time preprocessing algorithm for Π.

The notion of 1-safe reduction rules is crucial because numerous reduction rules used in the domain of (standard) kernelization can be either easily, or with very little effort, proved to be 1-safe. Therefore, when designing α-approximate kernels, it is a useful strategy to examine existing 1-safe reduction rules and either utilize them directly or design a 'relaxed' version which one can then prove to be α-safe for some $\alpha > 1$.

▶ **Definition 10** ([19]). Let $\alpha \geq 1$ be a real number. Let Π and Π' be two parameterized minimization problems. An *α-approximate polynomial parameter transformation* (α-appt for short) $\mathcal{A}$ from Π to Π' is a pair of polynomial time algorithms, called reduction algorithm $\mathcal{R}_{\mathcal{A}}$ and solution lifting algorithm. Given as input an instance (I, k) of Π the reduction algorithm outputs an instance (I', k') of Π' such that $k' = k^{\mathcal{O}(1)}$. The solution lifting algorithm takes as input an instance (I, k) of Π, the output instance $(I', k') = \mathcal{R}_{\mathcal{A}}(I, k)$ of Π', and a solution s' to the instance I' and outputs a solution s to (I, k) such that

$$\frac{\Pi(I, k, s)}{OPT_{\Pi}(I, k)} \leq \alpha \cdot \frac{\Pi'(I', k', s')}{OPT_{\Pi'}(I', k')}.$$

▶ **Definition 11** ([19]). Let $\alpha \geq 1$ be a real number. Let Π and Π' be two parameterized minimization problems. An *α-approximate compression* from Π to Π' is an α-appt $\mathcal{A}$ from Π to Π' such that $\text{size}_{\mathcal{A}}(k) = \sup\{|I'| + k' : (I', k') = \mathcal{R}_{\mathcal{A}}(I, k), I \in \Sigma^*\}$, is upper bounded by a computable function $g : \mathbb{N} \to \mathbb{N}$, where $\mathcal{R}_{\mathcal{A}}$ is the reduction algorithm in $\mathcal{A}$. We say that $\mathcal{A}$ is an α-approximate polynomial compression if g is a polynomial function. When we simply say that Π has an α-approximate compression, we mean that there is an α-approximate compression from Π to some language Π'.

▶ **Observation 12.** *Let $\alpha, \beta \geq 1$ be fixed real numbers and let Π and Π' be parameterized minimization problems. If there is an α-*appt *from Π to Π' and Π' has a β-approximate polynomial compression, then Π has an $(\alpha \cdot \beta)$-approximate polynomial compression.*

Steiner trees, set systems and degenerate graphs. Given a graph G, a set $R \subseteq V(G)$ whose vertices are called *terminals*, and a weight function $w : E(G) \to \mathbb{N}$, a *Steiner tree* is a subtree T of G such that $R \subseteq V(T)$, and the *cost* of a tree T is defined as $w(T) = \sum_{e \in E(T)} w(e)$. A *$k$-component* is a tree with at most k leaves which all coincide with a subset of terminals. A *k-restricted Steiner tree* $\mathcal{T}$ is a collection of k-components, such that the union of these components is a Steiner tree T. The cost of $\mathcal{T}$ is the sum of the costs of all the k-components in $\mathcal{T}$.

Let $\mathcal{F}$ be a set system over the universe $\mathcal{U}$. An element $e \in \mathcal{U}$ is said to *hit* a set $A \in \mathcal{F}$, if $e \in A$. Moreover, we say that a set $S \subseteq \mathcal{U}$ hits a set $A \in \mathcal{F}$ if $S \cap A \neq \emptyset$. A set $S \subseteq \mathcal{U}$ is a *hitting set* of $\mathcal{F}$ if it hits every set in $\mathcal{F}$. We say that $\mathcal{S} \subseteq \mathcal{F}$ is a *set cover* of $(\mathcal{F}, \mathcal{U})$, if $\bigcup_{S \in \mathcal{S}} S = \mathcal{U}$. Note that if there exists a set cover of $(\mathcal{F}, \mathcal{U})$, then there is one of size at most $|\mathcal{U}|$, as every vertex is in at least one set. Moreover, one can find a set cover of size at most $|\mathcal{U}|$ by greedily adding, in every step, a new set that covers an as yet uncovered vertex.

Let $\mathcal{H}$ be a fixed finite set of finite graphs. We denote by $d_{\mathcal{H}}$ the size of a largest graph in $\mathcal{H}$. For a graph G, we denote by $\mathcal{F}_{\mathcal{H}}(G)$ the following set system defined over the universe $V(G)$: $\mathcal{F}_{\mathcal{H}}(G) = \{S \subseteq V(G) \mid G[S] \text{ is isomorphic to some } H \in \mathcal{H}\}$. That is, $\mathcal{F}_{\mathcal{H}}(G)$ comprises precisely those subsets of $V(G)$ which induce a subgraph of G isomorphic to a graph in $\mathcal{H}$. Observe that for a fixed family $\mathcal{H}$ and a graph G, the set $\mathcal{F}_{\mathcal{H}}(G)$ can be computed in time $\mathcal{O}(|V(G)|^{d_{\mathcal{H}}})$. A set $S \subseteq V(G)$ is called a $\mathcal{H}$-hitting set of G if it is a hitting set for the family $\mathcal{F}_{\mathcal{H}}(G)$. A graph G is said to be d-degenerate if every subgraph of G has a vertex of degree at most d. It is well-known that a d-degenerate graph G has less than $d|V(G)|$ edges.

3 Approximate kernels for Connected $\mathcal{H}$-hitting set

In this section, we present our positive and negative results on the Connected $\mathcal{H}$-hitting set problem and its weighted variant. In what follows, we fix a family $\mathcal{H}$ and assume without loss of generality that for any distinct pair of graphs in $\mathcal{H}$, neither is a subgraph of the other.

3.1 The α-approximate kernel for Connected $\mathcal{H}$-hitting set

We begin by defining the parameterized optimization version of Conn-$\mathcal{H}$-HS. This is a minimization problem, where the optimization function is CHS : $\Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$ and defined as follows.

$$\mathrm{CHS}(G,k,S) = \begin{cases} \infty & \text{if } S \text{ is not a connected } \mathcal{H}\text{-hitting set in } G, \\ \min\{|S|, k+1\} & \text{otherwise.} \end{cases}$$

For the rest of Section 3.1, we define $\mathrm{OPT}(G,k) = \min_{S \subseteq V(G)} \mathrm{CHS}(G,k,S)$.

We split our approximate kernel for Conn-$\mathcal{H}$-HS into two steps. Let d denote the size of the largest graph in $\mathcal{H}$. First we compute a set D of size at most $k^{\mathcal{O}(d)}$ such that if a set S of size at most k is an $\mathcal{H}$-hitting set in $G[D]$, then S is an $\mathcal{H}$-hitting set in G. In the second step, we closely follow the idea of the approximate kernel for Steiner Tree (see [5, 19]) to bound the number of vertices outside D that we need to preserve to guarantee a 'good' connected set that hits all subgraphs in $G[D]$ isomorphic to a graph in $\mathcal{H}$.

Our starting point is the known kernel of size $k^{\mathcal{O}(d)}$ for the (not necessarily connected) $\mathcal{H}$-Hitting Set problem [6]. This kernel uses the *sunflower reduction rule* which is based on the classic sunflower lemma [14]. The sunflower reduction rule will also be a critical part of our approximate kernel and we begin by recalling the formal definition of sunflowers. Let $\mathcal{F}$ be a set system over the universe $\mathcal{U}$ and let $s \in \mathbb{N}$. A set $A_1, \ldots, A_s \in \mathcal{F}$ is called an *s-sunflower* if for every i, j such that $1 \leq i < j \leq s$, $A_i \cap A_j = \cap_{r=1}^{s} A_r$.

▶ **Proposition 13** ([14]). *Let $\mathcal{F}$ be a set system and d the size of a largest set in $\mathcal{F}$. If $|\mathcal{F}| > d!(k+1)^d$ $\mathcal{F}$, then $\mathcal{F}$ contains a $(k+2)$-sunflower. Moreover, it can be found in time polynomial in $|\mathcal{F}|$, k, and d.*

We now state and prove the following lemma, which is crucial for the correctness of the sunflower reduction rule. Although the following lemma is well-known, we state it in a way that is most convenient for our application.

▶ **Lemma 14.** *Let $\mathcal{U}$ be a universe of elements, $\mathcal{F}_1 \supset \mathcal{F}_2 \supset \cdots \supset \mathcal{F}_r$ be a family of set systems over $\mathcal{U}$ and $A_1, \ldots, A_{r-1} \subseteq \mathcal{U}$ such that for every $i \in \{1, \ldots, r-1\}$ the following holds: (a) $\mathcal{F}_{i+1} = \mathcal{F}_i \setminus A_i$ and (b) A_i is contained in a $(k+2)$-sunflower in $\mathcal{F}_i$. Then, if a set $S \subseteq \mathcal{U}$ of size at most k hits all sets in $\mathcal{F}_r$, then S also hits all sets in $\mathcal{F}_1$.*

We are now ready to formally describe the construction of the set D.

▶ **Lemma 15.** *Fix $\mathcal{H}$ and let $d = d_{\mathcal{H}}$. There exists a polynomial time algorithm that takes as input a graph G and integer k and outputs a set of vertices $D \subseteq V(G)$ of size at most $d \cdot d!(k+1)^d$ such that if a set S of size at most k is an $\mathcal{H}$-hitting set in $G[D]$, then S is an $\mathcal{H}$-hitting set in G.*

Due to this lemma, once we compute the set D, the *hitting* part of the Conn-$\mathcal{H}$-HS problem is taken care of and it is the *connectivity* which results in the hardness of standard kernelization. In other words, for every connected $\mathcal{H}$-hitting set S of a graph G of size at most k it follows from Lemma 15 that $D \cap S$ is also a $\mathcal{H}$-hitting set of G and the only role of vertices in $S - D$ is to connect the set of vertices in $D \cap S$. Such a situation could be handled relatively easily in the case of Connected Vertex Cover (see [19]) since the graph induced on $V(G) \setminus D$ is by definition an *independent set*. However, since we are dealing with an arbitrary family $\mathcal{H}$, we cannot rely on any structural consequences of a graph excluding $\mathcal{H}$; only the fact that the size of the largest graph in $\mathcal{H}$ is a fixed integer. A natural approach to providing connectivity between vertices in a graph is to construct a Steiner tree over a particular set of terminals. Unfortunately, since we do not know the set $S \cap D$ apriori, one would have to try and compute an appropriate Steiner tree for every possible subset of D of size at most $k-1$, as the set of terminals. Since this would be too expensive for us, we will try to preserve all necessary *approximate* Steiner trees. We begin by recalling the following result of Borchers and Du [4].

▶ **Proposition 16** ([4]). *For every $t \geq 1$, graph G, terminal set R, cost function $w : E(G) \to \mathbb{N}$, and Steiner tree T, there is a t-restricted Steiner tree $\mathcal{T}$ of cost at most $(1+\frac{1}{\lfloor \log_2 t \rfloor}) \cdot w(T)$.*

It follows from the proof of Borchers and Du [4] that if for every subset of R of size at most t, one were to preserve an *optimal* Steiner tree for this subset, then it is possible to construct a t-restricted Steiner tree of R of cost at most that of the tree $\mathcal{T}$ in Proposition 16 (see also [5, 19]). This fact will be used crucially in our algorithm.

▶ **Lemma 17.** *For every fixed $\epsilon > 0$, there exists a polynomial time algorithm that takes as an input a connected graph G and k and either correctly determines that G does not contain a connected $\mathcal{H}$-hitting set of size at most k or outputs an induced subgraph G' of G of size $\mathcal{O}(k^{d \cdot 2^{\frac{1}{\epsilon}}+1})$ such that:(1) if S is a connected $\mathcal{H}$-hitting set in G', then S is a connected $\mathcal{H}$-hitting set in G and (2) $\mathrm{OPT}(G', k) \leq (1+\epsilon) \cdot \mathrm{OPT}(G, k)$.*

Proof. The algorithm first executes the algorithm of Lemma 15 to obtain a set of vertices D of size at most $d \cdot d!(k+1)^d$ such that if a set S of size at most k is an $\mathcal{H}$-hitting set in $G[D]$, then S is an $\mathcal{H}$-hitting set in G. We fix a constant t such that $\frac{1}{\lfloor \log_2 t \rfloor} \leq \epsilon$. Now for every subset R of D of size at most t we fix a cost function assigning 1 to every edge and compute an *optimal* Steiner tree T_R for the set of terminals R using, for example, the Dreyfus-Wagner Algorithm [13]. It follows from [13] that this step takes time $\mathcal{O}(3^t |E(G)||V(G)|)$, which is polynomially bounded since ϵ is a fixed constant. If $|V(T_R)| \leq k$, then we *mark* the vertices of T_R. After we have computed T_R for every subset R (of size at most t) of D, we remove all unmarked vertices from G and denote the resulting graph by G'. We now claim that G' is the desired graph. It is easy to see that $|V(G')| \leq \sum_{i=1}^{t} \binom{|D|}{i} \cdot k = \mathcal{O}(k^{d \cdot t+1})$. Moreover, every connected $\mathcal{H}$-hitting set in G' is a $\mathcal{H}$-hitting set in $G[D]$ and hence it is also a connected $\mathcal{H}$-hitting set in G.

Note that if G' contains two different connected components A, B, then a shortest path with one endpoint in $A \cap D$ and the other in $B \cap D$ must have at least $k+1$ vertices. Otherwise, we would have marked such a path and A and B would not be distinct connected components of G'. Therefore, if both $A \cap D$ and $B \cap D$ contain a subgraph isomorphic to a graph in $\mathcal{H}$, then every connected $\mathcal{H}$-hitting set of G contains at least $k+1$ vertices and we may correctly return that G does not contain a connected $\mathcal{H}$-hitting set of size at most k.

Otherwise, for at most one component C of G', the graph $G[C \cap D]$ contains an induced subgraph isomorphic to a graph in $\mathcal{H}$. Since every $\mathcal{H}$-hitting set in $G[D]$ is a $\mathcal{H}$-hitting set in G, it follows that every *connected* $\mathcal{H}$-hitting set of $G[C]$ is also a connected $\mathcal{H}$-hitting set of G. Therefore, we assume in the following that G' is connected.

It remains for us to prove that $\mathrm{OPT}(G', k) \leq (1+\epsilon) \cdot \mathrm{OPT}(G, k)$. Observe that by the definition of the function OPT, it must be the case that $OPT(G, k), OPT(G', k) \leq k+1$. This is simply because $\mathrm{CHS}(G, k, V(G))$ and $\mathrm{CHS}(G', k, V(G'))$ are both bounded by $k+1$. Now, if it is the case that $\mathrm{OPT}(G, k) = k+1$, then $\mathrm{OPT}(G', k) \leq k+1 \leq (1+\epsilon) \cdot \mathrm{OPT}(G, k)$.

Therefore, we may assume that $\mathrm{OPT}(G, k) \leq k$. Let Q be an optimal connected $\mathcal{H}$-hitting set in G of size at most k. That is, $\mathrm{CHS}(G, k, Q) = OPT(G, k) = |Q|$. We denote $Q_D = Q \cap D$ and $Q_R = Q \setminus D$. Clearly, Q_D is a $\mathcal{H}$-hitting set in $G[D]$ and by our construction of D, it follows that Q_D is a $\mathcal{H}$-hitting set in G. Hence, if we consider the Steiner tree instance obtained by assigning every edge in G weight 1 and choosing Q_D as the set of terminals, any spanning tree T of $G[Q]$ must in fact be an optimal Steiner tree in G for the aforementioned weight function and terminal set Q_D. We invoke Proposition 16 to infer that there is a t-restricted Steiner tree $\mathcal{T}$ of cost at most $(1 + \frac{1}{\lfloor \log_2 t \rfloor}) \cdot (|Q| - 1)$. It remains to argue that we can reconstruct such a t-restricted Steiner tree $\mathcal{T}$ for Q_D using only the vertices in G'.

Consider a t-component C in $\mathcal{T}$ and let R be the set of terminals in C. Since $R \subseteq Q_D$, it implies that G' contains an optimal Steiner tree T_R for R. Moreover, C is a Steiner tree with R as the set terminals, hence $|T_R| \leq |C|$ and we can replace C by T_R in $\mathcal{T}$. Exhaustively repeating this argument we conclude that there is a t-restricted Steiner tree $\mathcal{T}'$ with set of terminals Q_D of cost no more than $(1 + \frac{1}{\lfloor \log_2 t \rfloor}) \cdot (|T| - 1)$, such that all k-components in $\mathcal{T}'$ use only marked vertices. Furthermore, (see paragraph following Proposition 16), the union of all t-components in $\mathcal{T}'$, denoted by $\bigcup \mathcal{T}'$, is indeed a Steiner tree. In particular, $\bigcup \mathcal{T}'$ is connected and contains all vertices in Q_D. Therefore, $\bigcup \mathcal{T}'$ is a connected $\mathcal{H}$-hitting set in G of size at most $(1 + \frac{1}{\lfloor \log_2 t \rfloor}) \cdot (|Q| - 1) + 1 \leq (1 + \epsilon)|Q|$. Since $|Q|$ is by definition the same as $\mathrm{OPT}(G, k)$, the lemma follows. ◀

▶ **Theorem 1.** *For every fixed $\epsilon > 0$, there is a $(1 + \epsilon)$-approximate polynomial kernel for* Connected $\mathcal{H}$-hitting set.

Proof. We begin by describing the reduction algorithm. We first invoke the algorithm of Lemma 17. If this algorithm concludes that G does not contain a connected $\mathcal{H}$-hitting set of size at most k, then we return the instance $(H, 0)$, where $H \in \mathcal{H}$. Otherwise, if this algorithm returns a graph G', then the reduction algorithm returns the instance (G', k). From Lemma 17 it follows that the size of the reduced instance is $\mathcal{O}(k^{d \cdot 2^{-\frac{1}{\epsilon}}})$.

We now describe the solution lifting algorithm as follows. Let S' be the given solution for (G', k). If S' is not a connected $\mathcal{H}$-hitting set in G', then the algorithm outputs $\emptyset$. If S' is a connected $\mathcal{H}$-hitting set in G', then the algorithm outputs S', if $|S'| \leq k$ and $V(G)$ otherwise. We denote by S the output of the solution lifting algorithm.

We now prove that this reduction algorithm and the solution lifting algorithm together constitute a $(1+\epsilon)$-approximate kernel. Note that if S' is not a connected $\mathcal{H}$-hitting set of G', then $\emptyset$ is also not a connected $\mathcal{H}$-hitting set of G and $\mathrm{CHS}(G', k', S') = \mathrm{CHS}(G, k, \emptyset) = \infty$. On the other hand, if $\mathrm{OPT}(G, k) = k + 1$, then it follows from Lemma 17 and the definition of the reduction algorithm that $\mathrm{OPT}(G', k') = k' + 1$. Therefore,

$$\frac{\mathrm{CHS}(G, k, V(G))}{\mathrm{OPT}(G, k)} = 1 \leq (1 + \epsilon) \cdot \frac{\mathrm{CHS}(G', k', S)}{\mathrm{OPT}(G', k')} = (1 + \epsilon)$$

Hence, we can assume that $\mathrm{OPT}(G, k) \leq k$ and the reduction algorithm returned the instance (G', k) such that G' is as in Lemma 17. Then either $|S'| \leq k$ and $S = S'$ or $|S'| \geq k + 1$ and $S = V(G)$. However, in both cases it holds that $\mathrm{CHS}(G, k, S) = \mathrm{CHS}(G', k, S')$. Moreover, from Lemma 17 it follows that $\mathrm{OPT}(G', k) \leq (1 + \epsilon) \cdot \mathrm{OPT}(G, k)$, implying the theorem. ◀

3.2 The lower bound for Weighted Connected $\mathcal{H}$-hitting set

In this section, we prove that in the presence of weights, the Connected $\mathcal{H}$-hitting set problem no longer admits an α-approximate kernel for *any* constant α. The parameterized optimization version of Weighted Connected $\mathcal{H}$-hitting set is formally defined via the function W-CHS : $\Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$ as follows: $\text{W-CHS}((G, w), k, S) = \infty$

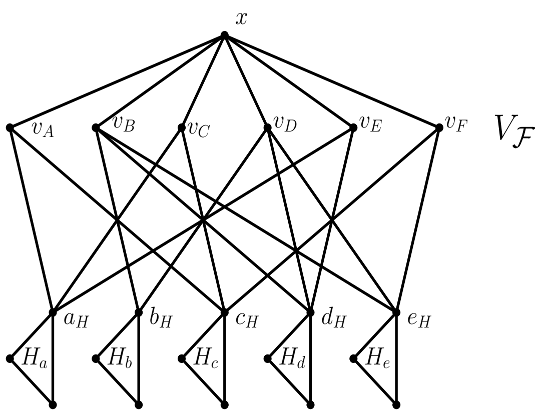

Figure 1 The graph output by our reduction algorithm starting from the SC/n instance $(\{A,B,C,D,E,F\},\{a,b,c,d,e\})$ where the sets are defined as $A=\{a,c\}$, $B=\{b,d,e\}$, $C=\{a,c\}$, $D=\{b,d,e\}$, $E=\{a,d\}$, and $F=\{c,e\}$. Here, $\mathcal{H}$ contains only a triangle.

if S is not a connected $\mathcal{H}$ − hitting set of size at most k and W-CHS$((G,w),k,S)=w(s)$ otherwise.

We prove our lower bound by giving a polynomial time reduction from a parameterized optimization version of the classic Set Cover problem such that an α-approximate polynomial kernel for Weighted Connected $\mathcal{H}$-hitting set would imply one for Set Cover, which would contradict the lower bound in [19]. Note that since we are proving a lower bound, it is sufficient to demonstrate *one* family $\mathcal{H}$ for which Weighted Conn-$\mathcal{H}$-HS does not admit approximate kernels. However, in the interest of extracting the strongest possible consequence of our reduction, we introduce the following definition.

▶ **Definition 18.** Let $\mathcal{H}$ be a fixed finite family of finite graphs. We say that $\mathcal{H}$ is *rigid* if there is a connected graph H in $\mathcal{H}$ and a vertex $v\in V(H)$ such that no graph $H'\in\mathcal{H}$ is a subgraph of H and no graph $H'\in\mathcal{H}$ is the disjoint union of connected components each of which is isomorphic to $H-v$.

▶ **Theorem 19.** *Let $\mathcal{H}$ be a fixed rigid family of graphs. Then, there is no α-approximate polynomial compression for* Weighted Conn-$\mathcal{H}$-HS *for any constant α unless* NP $\subseteq$ coNP/Poly *even if the weight function is restricted to* $\{0,1\}$.

Proof. We prove the theorem by giving a 1-approximate polynomial parameter transformation from SC/n to the Weighted Connected $\mathcal{H}$-hitting set problem. Recall that a polynomial parameter transformation consists of two algorithms, a reduction algorithm and a solution lifting algorithm. We describe a reduction algorithm that takes as input an instance $(\mathcal{F},\mathcal{U})$ of SC/n and outputs an instance (G,k,w) of Weighted Connected $\mathcal{H}$-hitting set such that $k=2|\mathcal{U}|+1$, $|G|\le 1+|\mathcal{F}|+d_{\mathcal{H}}|\mathcal{U}|$.

Reduction Algorithm. We construct G from $(\mathcal{F},\mathcal{U})$ as follows. The vertex set $V(G)$ is partitioned into sets $\{x\}\uplus V_{\mathcal{U}}\uplus V_{\mathcal{F}}$. Fix a graph $H\in\mathcal{H}$ which certifies the rigidity of $\mathcal{H}$. That is, there is a vertex $h^*\in V(H)$ such that no graph $H'\in\mathcal{H}$ is the disjoint union of connected components each of which is isomorphic to $H-h^*$. The set $V_{\mathcal{U}}$ induces in G, a disjoint copy H_u of H for every element $u\in\mathcal{U}$. We fix a special vertex $u_H\in H_u$ for every $u\in\mathcal{U}$. This vertex is the vertex of H_u corresponding to h^*. This is to ensure that after deleting u_H from each H_u, we do not still have a graph from $\mathcal{H}$ contained in $G[V_{\mathcal{U}}]$. The set $V_{\mathcal{F}}$ contains a vertex v_S for every set $S \in \mathcal{F}$. Finally, x is a vertex disjoint from $V_{\mathcal{U}} \cup V_{\mathcal{F}}$. The edge set of G is defined as follows $E(G) = \{xv_S | v_S \in V_{\mathcal{F}}\} \cup \{v_S u_H | u \in S\} \cup_{u \in \mathcal{U}} E(H_u)$. In other words, $E(G)$ contains beside the edges for every copy of H, an edge between x and every vertex in $V_{\mathcal{F}}$ and then an edge between a vertex $v_S \in V_{\mathcal{F}}$ corresponding to the set S and the previously fixed special vertex u_H in the copy of H corresponding to an element u, if and only if $u \in S$ (see Figure 1). Finally, the weight function $w : V(G) \to \{0, 1\}$ is defined as follows. We let $w(v) = 0$ if $v \in \{x\} \cup V_{\mathcal{U}}$ and $w(v) = 1$ otherwise. The *weight* of a set $Q \subseteq V(G)$ is defined as $\Sigma_{q \in Q} w(q)$. This completes the description of the reduction algorithm.

Solution Lifting Algorithm. The solution lifting algorithm is straightforward. Given a solution string T for the instance (G, k, w), if T is not a connected $\mathcal{H}$-hitting set of size at most k, then we return a spurious solution string for the instance $(\mathcal{F},\mathcal{U})$. Otherwise, we return the sets in $\mathcal{F}$ which correspond to $V_{\mathcal{F}} \cap T$.

We are now ready to prove that this is a 1-approximate polynomial parameter transformation. Observe that in order to do so, it is sufficient to prove the following claim.

▶ **Claim 20.** *For every $p \in \mathbb{N}$ there is a set cover of $(\mathcal{F},\mathcal{U})$ of size p if and only if there is a connected $\mathcal{H}$-hitting set of G with at most k vertices and weight exactly p.*

Proof. Suppose that $\mathcal{S}$ is a set cover of $(\mathcal{F},\mathcal{U})$. We can assume without loss of generality that $|\mathcal{S}| \leq |\mathcal{U}|$. We claim that $T = \{x\} \cup \{v_S | S \in \mathcal{S}\} \cup \{u_H | u \in \mathcal{U}\}$ is a weighted connected $\mathcal{H}$-hitting set of G of weight $|\mathcal{S}|$. As all vertices in $V_{\mathcal{F}}$ have weight 1 and all other vertices have weight 0, the weight of T is $|\mathcal{S}|$. Moreover, all vertices in $V_{\mathcal{F}}$ are adjacent to x and since $\mathcal{S}$ is a set cover, every vertex u_H is adjacent to a vertex v_S for a set $S \in \mathcal{S}$ that contains u. Finally every connected component of $G - T$ is either a vertex or a graph isomorphic to $H - h^*$. Since $\mathcal{H}$ is rigid, we conclude that $G - T$ does not contain a graph in $\mathcal{H}$. Since $\mathcal{S}$ has size at most $|\mathcal{U}|$, the size of T is bounded by $2|\mathcal{U}| + 1$ which is precisely k.

In the converse direction suppose that T is a connected $\mathcal{H}$-hitting set of G of weight p. Since the only vertices with non-zero weights lie in $V_{\mathcal{F}}$ and they all have weight 1, we infer that $|V_{\mathcal{F}} \cap V(T)| = p$. We claim that $\mathcal{S} = \{S | v_S \in V(T) \cap V_{\mathcal{F}}\}$ is a set cover of $(\mathcal{F},\mathcal{U})$. Observe that since T is a $\mathcal{H}$-hitting set, it must be the case that for every $u \in \mathcal{U}$, T contains a vertex from H_u. Since T also contains at least one vertex of $V_{\mathcal{F}}$ (under the simple assumption that $|\mathcal{U}| > 1$), and only u_H is adjacent to a vertex outside H_u, it follows that $u_H \in V(T)$. Moreover, u_H is adjacent only to vertices in H_u or in $V_{\mathcal{F}}$. Therefore, u_H is adjacent to a vertex $v_S \in V_{\mathcal{F}} \cap V(T)$ for a set $S \in \mathcal{F}$. This implies that the element u is covered by the set $S \in \mathcal{S}$, completing the proof of the claim and the proof of the lemma. ◀

◀

4 Interpolating kernels for Dominating Set on d-degenerate graphs

This section is devoted to Theorem 2, i.e., the approximate kernels interpolating between two known kernels with respect to their accuracy-size tradeoff.

▶ **Proposition 21.** [20] Dominating Set *has a kernel of size $\mathcal{O}((d + 2)^{2(d+2)} k^{2(d+1)^2})$ on d-degenerate graphs.*

▶ **Definition 22.** The parameterized optimization version of Dominating Set is defined via the function $\mathrm{DS} : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{R} \cup \{\pm\infty\}$ as follows:

$$\mathrm{DS}(G,k,S) = \begin{cases} \infty & \text{if } S \text{ is not a dominating set of } G, \\ \min\{|S|, k+1\} & \text{otherwise.} \end{cases}$$

For the rest of this section, we define $\mathrm{OPT}(G,k) = \min_{D \subseteq V(G)} \mathrm{DS}(G,k,D)$.

The kernelization algorithm of Philip et al. [20] can be seen to be a strict 1-approximate polynomial kernel and forms the starting point of our sequence of approximate kernels. We give here a slightly different description of this kernel (in particular of its analysis) so as to better serve our purposes. First of all, we will be working with a “colored" version of the problem where the vertices of the input graph are partitioned into two sets – the set of red vertices R and the set of blue vertices B – and the goal is to find a subset of at most k vertices of any color that dominates all red vertices. That is, a set $S \subseteq R \cup B$ with $|S| \leq k$ such that for every $v \in R$ we have $N[v] \cap S \neq \emptyset$. Clearly every instance of Dominating Set can be reduced to the colored variant by coloring all the vertices red. For presentation purposes, we will refer to the colored version as Dominating Set and instances of this problem are of the form (G, B, R, k) where B and R denote the set of blue and red vertices respectively. The functions $\mathrm{DS}(G,k,S)$ and $\mathrm{OPT}(G,k)$ are now represented as $\mathrm{DS}(G,B,R,k,S)$ and $\mathrm{OPT}(G,B,R,k)$ with the natural extended definitions. Furthermore, since edges between vertices in B are irrelevant with respect to the domination of R, we may assume without loss of generality that B is an independent set. Philip et al. [20] devised the following reduction rule and their proof of correctness of the rule also shows that it is in fact 1-safe.

Let (G,B,R,k) be the given instance of Dominating Set. For $i \in \{0,\dots,d\}$: If there exists a set of $d+1-i$ vertices $X \subseteq R \cup B$ which have at least $k^i(d+1)$ common red neighbors $Y \subseteq R$, then remove the edges between X and Y, color all vertices in Y blue, and add $k+1$ new red vertices that are each connected to all vertices in X and no other vertex in G. The parameter remains k.

Henceforth, we assume that Reduction Rule 4 does not apply on the given instance of Dominating Set. Philip et al. [20] showed that if Reduction Rule 4 does not apply on the instance (G,B,R,k), then every vertex in G has at most $k^d(d+1)$ red neighbors, leading to the following observation.

▶ **Lemma 23.** *If Reduction Rule 4 does not apply on the instance (G,B,R,k), then either $|R| \leq k^{d+1}(d+1)$ or $\mathrm{OPT}(G,B,R,k) = k+1$.*

Due to Lemma 23, we may assume that $|R| \leq k^{d+1}(d+1)$. The following standard *twin reduction* rule can be easily seen to be 1-safe.

If $b_1, b_2 \in B$ are two non-adjacent vertices such that $N(b_1) = N(b_2)$, we remove b_1 from G.

In the following, for every $i \in \{0,\dots,d\}$, we let B_i denote the set of blue vertices which have exactly i red neighbors and let $B_{>d}$ denote the set of blue vertices which have at least $d+1$ red neighbors. We now prove the following bound on the size of each of these sets.

▶ **Lemma 24.** *Let (G,B,R,k) be an instance of* Dominating Set *on which Reduction Rule 4 and Reduction Rule 4 do not apply. Then, $|B_{>d}| \leq d|R|$, and $|B_i| \leq |R|^i$ for each $i = 0,\dots,d$.*

Observe that since we have only applied 1-safe reduction rules, the instance obtained after the exhaustive application of Reduction Rule 4 and Reduction Rule 4 is a strict 1-approximate kernel and due to Lemma 24, the result is a strict 1-approximate kernel of size $k^{\mathcal{O}(d^2)}$. This is the kernel of Philip et al. [20] and henceforth we refer to instances of Dominating Set on which this preprocessing has been executed, as *reduced instances* and assume without loss of generality that the input has size bounded by $k^{\mathcal{O}(d^2)}$. We will now introduce a 'lossy reduction rule' to reduce the size of our kernel further at the cost of transforming it into a $\lceil \frac{d}{\rho} \rceil$-approximate kernel.

▶ **Lemma 25.** *Let $d, \rho \in \mathbb{N}$ be fixed integers such that $\rho < d$. There is an algorithm that, given a reduced instance (G, B, R, k) of* Dominating Set *runs in polynomial time and returns an instance (G', B', R', k) such that (a) $|V(G')| \leq k^{\mathcal{O}(\rho d)}$, (b) if S dominates R' in G', then S dominates R in G and (c) $\mathrm{OPT}(G', B', R', k) \leq \lceil \frac{d}{\rho} \rceil \cdot \mathrm{OPT}(G, B, R, k)$.*

Proof. Let B^* denote an auxiliary set of blue vertices which is initially empty. For each subset of ρ red vertices $R_0 \subseteq R$ we find a blue vertex $b \in B$ (if one exists) with $R_0 \subseteq N(b)$, and add it to our auxiliary set B^*. At the end of this procedure, we define the graph G' to be the subgraph of G induced by $R \cup B_0 \cup \cdots \cup B_\rho \cup B_{>d} \cup B^*$, $B' = B \cap V(G')$, and $R' = R$.

Recall that $|R| = \mathcal{O}(k^{d+1})$, and so there are $\binom{|R|}{\rho} = \mathcal{O}(k^{\rho(d+1)})$ subsets R_0. Thus, $|B^*| = \mathcal{O}(k^{\rho(d+1)})$. Moreover, $|B_0 \cup \cdots \cup B_\rho| = \mathcal{O}(k^{d\rho})$ according to Lemma 24. Therefore, $|V(G')|$ is bounded by $k^{\mathcal{O}(\rho d)}$ as required and the time required to compute G' is bounded polynomially in $|V(G)|$. We now proceed to the remaining two statements. Since $R' = R$ and G' is a subgraph of G, it follows that any set S which dominates all vertices of R' in G', also dominates all vertices of R in G. Hence, it only remains to prove the second statement.

Let S be an optimal solution for G. That is, $\mathrm{OPT}(G, B, R, k) = |S|$. We now construct a solution S' for G' as follows. We begin by setting $S' = S \cap V(G')$. Note that S' includes all vertices of $R \cap S$ since $R \subseteq V(G) \cap V(G')$. Consider now a blue vertex $b \in S \setminus V(G')$, and let $R(b)$ denote the set of red neighbors of b. Then $\rho + 1 \leq |R(b)| \leq d$ by the construction of G'. Moreover, for any subset of ρ vertices in $R(b)$, there is a vertex of B' in $V(G')$ which dominates these ρ vertices. Thus, we can replace b with at most $\lceil \frac{d}{\rho} \rceil$ vertices of B' in $V(G')$ and still dominate $R(b)$. Therefore, applying this switch for each $b \in S \setminus V(G')$, we obtain a solution S' for G' with $|S'| \leq \lceil \frac{d}{\rho} \rceil |S|$. This implies that $\mathrm{OPT}(G', B', R', k) \leq \lceil \frac{d}{\rho} \rceil \cdot \mathrm{OPT}(G, B, R, k)$, completing the proof of the lemma. ◀

From Lemma 25 it immediately follows that Dominating Set on d-degenerate graphs has $\lceil \frac{d}{\rho} \rceil$-approximate compression to the colored version of the problem. Theorem 2 then follows by gadgeteering similar to that used by Philip et al. [20]. Note that any graph that excludes K_h as a minor also excludes K_h as a topological minor (see [9] for a formal definition of minors and topological minors). Furthermore, it is known that any graph that does not contain K_h as a minor (topological minor) is d-degenerate where $d = \mathcal{O}(h^2)$ ($d = \mathcal{O}(h\sqrt{\log h})$ respectively) [2], giving us the following corollary.

▶ **Corollary 26.** *Let $\rho, h \in \mathbb{N}$. Then,* Dominating Set *on graphs excluding K_h as a minor (topological minor) has a $\mathcal{O}(\frac{h^2}{\rho})$-approximate kernel ($\mathcal{O}(\frac{h\sqrt{\log h}}{\rho})$-approximate kernel) of size $k^{\mathcal{O}(\rho h^2)}$ ($k^{\mathcal{O}(\rho h \sqrt{\log h})}$ respectively).*

5 Conclusions

Our work on the Connected $\mathcal{H}$-Hitting Set problem adds another interesting data point to the study of preprocessing for problems with connectivity constraints. We have also initiated the study of accuracy-size tradeoffs for problems which already have polynomial kernels, via the design of a sequence of kernels capturing the gradient of the kernel-size with respect to the accuracy or approximation factor. Our results point to a few interesting questions for future research.

- Are there other connectivity-constrained problems which do not admit polynomial kernels but admit α-approximate kernels?
- Is it possible to obtain *meta-theorems* characterizing or providing at least a sufficiency condition for connectivity-constrained problems to admit α-approximate kernels?
- Is it possible to refine the interpolation (Theorem 2) by presenting a sequence of kernels between the d^2-approximate kernel of constant size and our d-approximate kernel of size $k^{\mathcal{O}(d)}$?

References

1 Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.

2 Noga Alon and Shai Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica*, 54(4):544–556, 2009. doi:10.1007/s00453-008-9204-0.

3 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.

4 Al Borchers and Ding-Zhu Du. The k-steiner ratio in graphs. In Frank Thomson Leighton and Allan Borodin, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 641–649. ACM, 1995.

5 Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013.

6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

7 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

8 Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *The Computer Journal*, 51(3):292–302, 2008.

9 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

10 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Transactions on Algorithms*, 11(2):13:1–13:20, 2014.

11 Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

12 Pål Grønås Drange, Markus Sortland Dregi, Fedor V. Fomin, Stephan Kreutzer, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, Felix Reidl, Fernando Sánchez Villaamil, Saket Saurabh, Sebastian Siebertz, and Somnath Sikdar. Kernelization and sparseness: the case of dominating set. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 31:1–31:14, 2016.

13 S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

14 P. Erdös and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, s1-35(1):85–90, 1960.

15 Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36:281–309, 2006.

16 Mark Jones, Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Ondrej Suchý. Parameterized complexity of directed steiner tree on sparse graphs. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 671–682, 2013.

17 Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113, 2014.

18 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization–preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161. Springer, 2012.

19 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 224–237. ACM, 2017.

20 Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Trans. Algorithms*, 9(1):11:1–11:23, 2012.