Sampling with a Black Box: Faster Parameterized Approximation Algorithms for Vertex Deletion Problems

Barış Can Esmer[*1] and Ariel Kulik[†2]

[1]CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. baris-can.esmer@cispa.de
[2]Computer Science Department, Technion, Haifa, Israel. kulik@cs.technion.ac.il

Abstract

In this paper we introduce *Sampling with a Black Box*, a generic technique for the design of parameterized approximation algorithms for vertex deletion problems (e.g., Vertex Cover, Feedback Vertex Set, etc.). The technique relies on two components:

- A *Sampling Step.* A polynomial time randomized algorithm which given a graph G returns a random vertex v such that the optimum of $G \setminus \{v\}$ is smaller by 1 than the optimum of G with some prescribed probability q. We show such algorithms exists for multiple vertex deletion problems.
- A Black Box algorithm which is either an exact parameterized algorithm or a polynomial time approximation algorithm.

Our technique combines these two components together. The sampling step is applied iteratively to remove vertices from the input graph, and then the solution is extended using the black box algorithm. The process is repeated sufficiently many times so that the target approximation ratio is attained with a constant probability. The main novelty of our work lies in the analysis of the framework and the optimization of the parameters it uses.

We use the technique to derive parameterized approximation algorithm for several vertex deletion problems, including Feedback Vertex Set, d-Hitting Set and ℓ-Path Vertex Cover. In particular, for every approximation ratio $1 < \beta < 2$, we attain a parameterized β-approximation for Feedback Vertex Set which is faster than the parameterized β-approximation of [Jana, Lokshtanov, Mandal, Rai and Saurabh, MFCS 23']. Furthermore, our algorithms are always faster than the algorithms attained using Fidelity Preserving Transformations [Fellows, Kulik, Rosamond, and Shachnai, JCSS 18'].

1 Introduction

A vast body of research has been dedicated to basic vertex deletion problems such as Vertex Cover, 3-Hitting Set and Feedback Vertex Set. In these problems, the objective is to delete a minimum cardinality set of vertices from the input (hyper-)graph so that the remaining (hyper-)graph satisfies a specific property (edge-free, cycle-free, etc.). As many of these problems are NP-hard, multiple algorithmic results focus on either polynomial time approximations or exact

[*]The author is part of Saarbrücken Graduate School of Computer Science, Germany.

[†]This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 852780-ERC (SUBMODULAR).

parameterized algorithms. In between these two classes of algorithms lies the class of *parameterized approximation algorithms*. These algorithms aim to provide approximation ratios which cannot be attained in polynomial time. They operate within a parameterized running time, which is faster than the exact, parameterized state-of-the-art.

In this paper we explore how existing exact parameterized algorithms and polynomial time approximation algorithms can be used together with *sampling steps* to derive efficient parameterized approximation algorithms. Informally, a sampling step with success probability $q \in (0, 1)$ is a polynomial time algorithm which, given an input graph $G = (V, E)$, returns a random vertex $v \in V$. The vertex v should satisfy, with probability q or more, that removing v from G reduces its optimum (i.e., the number of vertices one needs to remove from the graph for it to satisfy the property) by 1. As we show in this paper, such algorithms can be easily obtained for various vertex deletion problems.

Our technique, *Sampling with a Black Box*, applies the sampling step t times and subsequently uses the existing parameterized/approximation algorithms to complete the solution. The whole process is executed sufficiently many times so that a β-approximate solution is found with a constant probability. The main novelty of our work lies in the analysis of the framework, involving tail bounds for binomial distribution and optimization of the parameters used by the technique.

Sampling with a Black-Box is applicable to a wide collection of vertex deletion problems. We show sampling steps exist for every vertex deletion problems, for which the property can be described by a finite set of forbidden vertex induced hypergraphs, such as d-Hitting Set, ℓ-Path Vertex Cover and Directed Feedback Vertex Set on Tournaments. We further provide sampling steps for Feedback Vertex Set (FVS) and Pathwidth One Vertex Deletion, where the set of forbidden hypergraphs is infinite. Moreover, even though our setting doesn't explicitly allow it, the results developed in this paper also apply to some problems in the directed graph setting. In particular, we show that there exists a sampling step for Directed Feedback Vertex Set on Tournaments. Thus, the technique is applicable for each of these problems.

We compare Sampling with a Black Box to existing benchmarks.

- In [28], Jana, Lokshtanov, Mandal, Rai, and Saurabh developed a parameterized β-approximation for Feedback Vertex Set (FVS) for every $1 < \beta < 2$. The objective in FVS is to remove a minimum number of vertices from an undirected graph, so the remaining graph does not contain cycles. Their approach relies on the same ingredients as ours: utilize the state of art parameterized and approximation algorithms in conjunction with a variant of the well known randomized branching rule of Becker et al. [3].

  Similar to [28], we utilize the randomized branching rule of [3] to derive a sampling step with success probability $\frac{1}{4}$. We use Sampling with a Blackbox together with this sampling step to attain a parameterized β-approximation for Feedback Vertex Set, for every $1 < \beta < 2$. Though we use the same core principles, we attain a faster running time for *every* approximation ratio between 1 and 2 – see comparison in Figure 1.

  The improved running time stems from our tighter analysis and careful selection of parameters, which also provides flexibility in the design of the sampling steps. For example, to attain a parameterized 1.1-approximation for FVS, the authors of [28] use ideas from [3] to derive a simple algorithm which randomly picks an *edge* in the graph G such that at least one of its endpoints is in a minimum solution with probability $\frac{1}{2}$. If the selected edge satisfies this property we refer to it as *correct*. In their scheme, every time an edge is picked, both its endpoints are added to the solution, reducing the optimum by one while increasing the solution size by two, assuming the picked edge is correct.

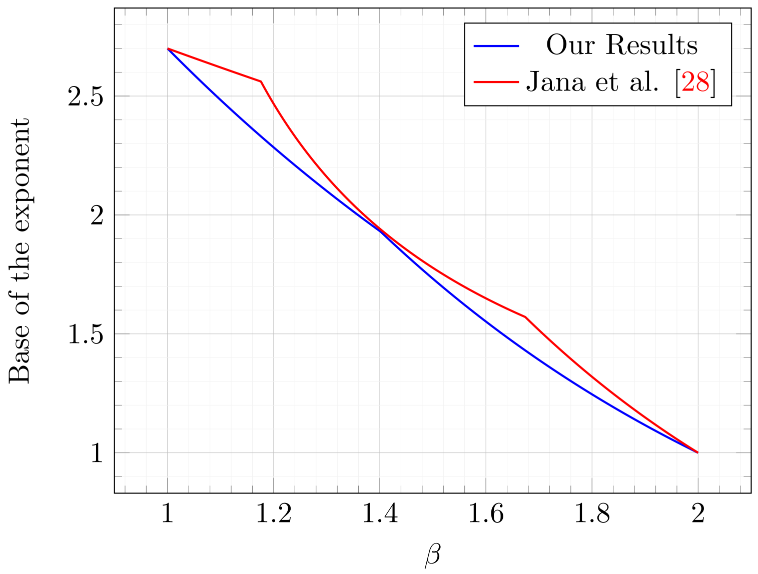

Figure 1: Comparison of the running times for FEEDBACK VERTEX SET. The x-axis corresponds to the approximation ratio, while the y-axis corresponds to the base of the exponent in the running time. A dot at (β, c) means that there is a parameterized β-approximation for FVS in time $c^k \cdot n^{\mathcal{O}(1)}$.

The analysis in [28] only considers the case in which the picked edges are *always* correct. This allows for a simple analysis and renders the parameter selection straightforward. Keeping our focus on a 1.1-approximation, the algorithm has to pick $0.1 \cdot k$ edges before it invokes the state-of-art parameterized algorithm with the parameter $k' = 0.9 \cdot k$. The success probability of the above – the probability that all picked edges are correct and therefore a 1.1-approximate solution is attained - is $0.5^{0.1 \cdot k}$, and the running time is $c^{0.9 \cdot k} \cdot n^{\mathcal{O}(n)}$, where $c = 2.7$ is the running time of the best known exact parameterized algorithm for FVS [33]. This leads to a running time of $2^{0.1 \cdot k} \cdot c^{0.9 \cdot k} \approx 2.62^k$.

A major limiting factor in the analysis of [28] is the focus on the event in which *all* picked edges are correct. For example, to attain a parameterized 1.1-approximation one can consider picking $0.09k$ edges (and add both endpoints to the solution), and then invoke the exact parameterized algorithm with $k' = 0.92k$. Now, this procedure finds a 1.1-approximate solution if $0.08k$ (or more) of the picked edges are correct. A careful analysis shows that the probability of such event is $\approx 0.708^{0.09 \cdot k} \approx 0.969^k$. By repeating this procedure $\frac{1}{0.969^k}$ times, a 1.1-approximate solution is found with a constant probability. The overall running time is $\left(\frac{c}{0.969}\right)^k \cdot n^{\mathcal{O}(1)} \approx 2.57^k \cdot n^{\mathcal{O}(1)}$, which is already an improvement over [28].

The above example illustrates the power of a more flexible analysis and a careful selection of parameters such as the number of sampled edges. Furthermore, our approach allows for more powerful sampling steps- instead of picking both endpoints of the selected edges, we can select one at random. Intuitively, the benefit of sampling one vertex at a time is that it increases the *variance* of the number vertices selected from the optimum, therefore raising the probability of the rare event the analysis is focused on. This change leads to a further improvement to the running time. Overall, we attained a parameterized 1.1-approximation for FVS in time $2.483^k \cdot n^{\mathcal{O}(1)}$.

- In [23], Fellows, Kulik, Rosamond and Shachnai provided a generic technique, called Fidelity

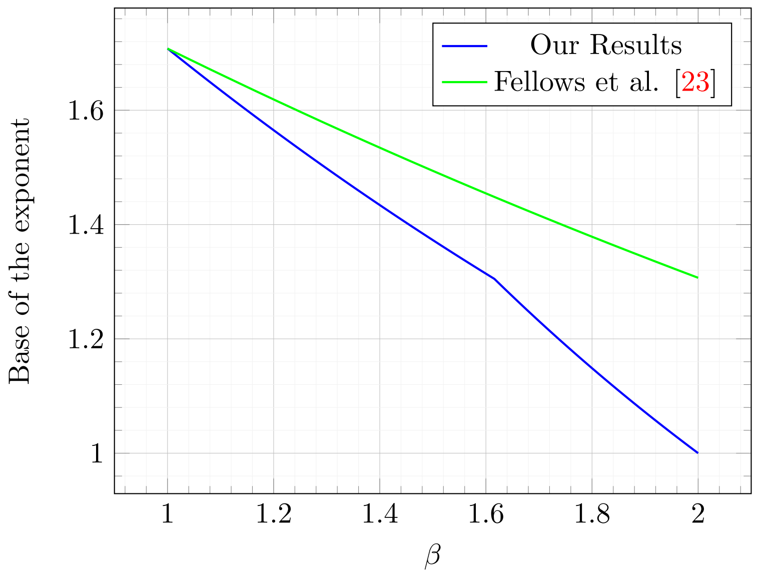

Figure 2: Comparison of the running times for 3-Path Vertex Cover. The x-axis corresponds to the approximation ratio, while the y-axis corresponds to the base of the exponent in the running time. A dot at (β, c) means that there is a parameterized β-approximation for `FVS` in time $c^k \cdot n^{\mathcal{O}(1)}$.

Preserving Transformations, which can be applied for every vertex deletion problem in which the property can be described by a finite set of forbidden vertex induced subgraphs. Assuming the maximum number of vertices in a forbidden induced subgraph is η, and the problem has an exact parameterized $c^k \cdot n^{\mathcal{O}(1)}$ algorithm, the technique of [23] yields a β-approximation in time $c^{\frac{\eta-\beta}{\eta-1} \cdot k} \cdot n^{\mathcal{O}(1)}$.[1] We prove that the running time we obtain for every problem in the class is always faster than the running time attained by [23], for every $1 < \beta < \eta$. Figure 2 provides a comparison between the running times we attain and those of [23] for 3-Path Vertex Cover, in which the objective is to remove vertices from a graph so the remaining graph does not have a path of 3 vertices.

1.1 Related Work

The field of parameterized approximation aims to derive approximation algorithms which run in parameterized running time. In the classic setting, the considered problem is one which is not expected to have an exact parameterized algorithm, and further has a hardness of approximation lower bound which indicates that a polynomial time α-approximation is unlikely to exist, for some $\alpha > 1$. In such a setting, the objective is to derive a β-approximation algorithm that runs in parameterized running time for $\beta < \alpha$, or to show that such an algorithm cannot exist (subject to common complexity assumptions). In recent years, there has been a surge of breakthrough results in the field, providing both algorithms (e.g., [32, 25, 20, 39, 37, 4, 1, 19, 27]) and hardness of approximation results (e.g., [26, 34, 47, 15, 6, 40, 29, 35, 16]).

Within the broad field of parameterized approximations, a subset of works consider problems which are in FPT. For problems in FPT, there is always a parameterized β-approximation algorithm for all $\beta \geq 1$, as the exact algorithm is also an approximation algorithm. Therefore, the main focus

[1]The result in [23] was stated for Vertex Cover and 3-Hitting Set, but can be easily generalized to the stated result.

of these works is on the *trade-off* between approximation and running time.

A prominent approach to attain such parameterized-approximation algorithms relies on using existing exact parameterized algorithms as black-box. In [8] the authors showed that an exact parameterized $c^k \cdot n^{\mathcal{O}(1)}$ algorithm for Vertex Cover implies a parameterized β-approximation in time $c^{(2-\beta)k}$. The same running time has been attained for Vertex Cover by Fellows et al. [23] through a more generic framework which can be applied to additional problems.

Other works focused on directly designing parameterized approximation algorithms. In [10, 9] Brankovic and Fernau provided parameterized approximation algorithms for Vertex Cover and 3-Hitting Set. The designed algorithms are branching algorithms which involve branching rules aimed for approximation and interleave approximative reduction rules. The works provide, among others, a parameterized 1.5-approximation for Vertex Cover in time $1.0883^k \cdot n^{\mathcal{O}(1)}$ and a parameterized 2-approximation for 3-Hitting Set in time $1.29^k \cdot n^{\mathcal{O}(1)}$.

In [30], Kulik and Shachnai showed that the use of randomized branching can lead to significantly faster parameterized-approximation algorithms. In particular, they attained a parameterized 1.5-approximation for Vertex Cover in time $1.017^k \cdot n^{\mathcal{O}(1)}$ and a parameterized 2-approximation for 3-Hitting Set in time $1.0659^k \cdot n^{\mathcal{O}(1)}$. The idea in randomized branching is that the algorithm picks one of the branching options at random. Subsequently, a good approximation is attained as long as the randomly picked options are not too far from the correct options. The branching rules used by [30] are fairly involved in comparison to the sampling steps used by this paper and their analysis required a non-trivial mathematical machinery.

While [30] showed that randomized branching is a powerful technique for the design of parameterized-approximation algorithms, it has only been applied to a limited set of problems. In [30], the authors provided applications of the technique for Vertex Cover and 3-Hitting Set. Furthermore, for approximation ratios close to 1, the randomized-branching algorithms are inferior to the exact algorithms in terms of running times (or to the combination of those with [23]). A central goal of this paper is to overcome some of these difficulties: Sampling with a Black Box harnesses the power of randomized branching, it is applicable for a wide set of problems, and always improves upon the running times attained using the best known exact algorithms in conjunction with [23].

As already mentioned, in [28], the authors developed parameterized approximation algorithms for `FVS` using the combination of randomized branching and existing algorithms as black-box. Their work indeed served as a motivation for this paper. We point out that the approach in [28] is restricted for `FVS` and the resulting running times are inferior to ours.

The concept of randomly sampling a partial solution, which is subsequently extended using a parameterized algorithm, is central to the *monotone local search* technique of Fomin, Gaspers, Lokshtanov and Saurabh [24]. Later variants of this technique are designed to obtain exponential time approximation algorithms [21, 22, 13]. They also use the same basic argument, which states that sampling a partial solution not too far from the optimum suffices to attain an approximate solution with high probability.

Organization Section Section 2 gives several standard definitions used throughout the paper. In Section 3 we formally state the results of the paper. Section 4 lists applications of the technique for several problems. Additional applications are given in Section 8. Section 5 provide our main algorithm together with its proof of correctness. In Section 6 we provide a simpler formula for the running time Sampling with a Black Box for several cases. In Section 7 we derive sampling steps for several problems. Section 9 provides the proof that the running time of Sampling with a Black Box is faster than Fidelity Preserving Transformations. Finally, we discuss our results in Section 10. For the sake of presentation, we include the proofs of some claims in the appendix.

2 Preliminaries

In this section, we present several standard concepts and notations used throughout this paper.

Graph Notations. Given an hypergraph $G = (V, E)$ we use $V(G) = V$ and $E(G) = E$ to denote the sets of vertices and hyperedges of the graph (respectively).

Vertex Induced Subhypergraphs and Vertex Deletion. Given a a hypergraph $G = (V, E)$ and a subset of vertices $U \subseteq V$, the *vertex induced subhypergraph* of G and U is the hypergraph $G[U] = (U, E')$ where $E' := \{e \in E \mid e \subseteq U\}$. We also define the *vertex deletion* of U from G by $G \setminus U = G[V \setminus U]$. For a single vertex $u \in V(G)$ we use the shorthand $G \setminus v = G \setminus \{v\}$.

Hypergraph Properties. A *hypergraph property* is a set Π of hypergraphs such that for every $G \in \Pi$ and G' isomorphic to G it holds that $G' \in \Pi$ as well. A hypergraph property Π is called *hereditary* if for every $G \in \Pi$ and $U \subseteq V(G)$ it holds that $G[U] \in \Pi$ as well. Furthermore, we say that Π is polynomial-time decidable if given a hypergraph G we can decide in polynomial-time whether $G \in \Pi$ or not. We assume that all hypergraph properties discussed in this paper are hereditary and polynomial-time decidable; that is by saying the Π is a hypergraph property we also mean that it is hereditary and polynomial-time decidable.

A closed set of hypergraph. We say that a set $\mathcal{G}$ of hypergraph is *closed* if every vertex induced subhypergraph of a graph in $\mathcal{G}$ is also in $\mathcal{G}$. That is, for every $G \in \mathcal{G}$ and $S \subseteq V(G)$ it also holds that $G[S] \in \mathcal{G}$.

Kullback-Leibler Divergence Given two number $a, b \in [0, 1]$, the *Kullback-Leibler divergence* of a and b is

$$\mathcal{D}\left(a \,\|\, b\right) = a \cdot \ln\left(\frac{a}{b}\right) + (1 - a) \cdot \ln\left(\frac{1-a}{1-b}\right).$$

We follow that standard convention that $0 \cdot \ln 0 = 0 \cdot \ln \frac{0}{x} = 0$, which implies

$$\mathcal{D}\left(1 \,\|\, b\right) = 1 \cdot \ln\left(\frac{1}{b}\right) + (1 - 1) \cdot \ln\left(\frac{1-1}{1-b}\right) = -\ln(b). \tag{1}$$

3 Our Results

In this section we formally state our results, starting with some formal definitions. The definitions of vertex deletion problems and parameterized approximation algorithms are given in Section 3.1. Section 3.2 provides the definition of sampling steps. Then, we state our main result in Section 3.3. The comparison of our results to those of [23] is finally given in Section 3.4.

3.1 Vertex Deletion Problems

The focus of this paper is the class of $(\mathcal{G}, \Pi)$-`Vertex Deletion` problems. For any hypergraph property Π and a closed set of hypergraphs $\mathcal{G}$, the input for $(\mathcal{G}, \Pi)$-`Vertex Deletion` ($(\mathcal{G}, \Pi)$-`Del` for short) is a hypergraph $G \in \mathcal{G}$. A *solution* is a subset of vertices $S \subseteq G(V)$ such that $G \setminus S \in \Pi$. We use $\texttt{SOL}_{\Pi}(G) = \{S \subseteq G(V) \mid G \setminus S \in \Pi\}$ to denote the set of all solutions. The objective is to find a smallest cardinality solution $S \in \texttt{SOL}_{\Pi}(G)$. In the decision version of the problem, we are given a hypergraph $G \in \mathcal{G}$, an integer $k \geq 0$ and we are asked whether there exists a set $S \in \texttt{SOL}_{\Pi}(G)$ such that $|S| \leq k$.

The family of $(\mathcal{G}, \Pi)$-`Vertex Deletion` problems include many well known problems. Some notable examples are:

- Vertex Cover. In this case $\mathcal{G}$ corresponds to graphs, i.e. hypergraphs with edge cardinality exactly 2. The hypergraph property Π consists of all edgeless graphs.
- Vertex Cover on graphs of degree at most 3. Similar to the case above, in this case $\mathcal{G}$ corresponds to graphs of degree at most 3. Π again consists of all edgeless graphs.
- d-Hitting Set. Similar to Vertex Cover, in this case $\mathcal{G}$ corresponds to hypergraphs with edge cardinality exactly d. Π also consists of all edgeless hypergraphs.
- Feedback Vertex Set. In this case $\mathcal{G}$ is the set of all graphs and Π is the set of graphs that have no cycles.

We use $\texttt{OPT}_{\mathcal{G},\Pi}(G)$ to denote the size of an optimal solution for $G \in \mathcal{G}$ with respect to the $(\mathcal{G}, \Pi)$-`Vertex Deletion` problem. That is, $\texttt{OPT}_{\mathcal{G},\Pi}(G) = \min \left\{ |S| \,\middle|\, S \in \texttt{SOL}_{\Pi}(G) \right\}$. If $\mathcal{G}$ and Π are known by context we use `OPT` instead of $\texttt{OPT}_{\mathcal{G},\Pi}$.

Our goal is to develop *parameterized approximation* algorithms for $(\mathcal{G}, \Pi)$-`Vertex Deletion` problems, where the parameter is the solution size. Such algorithms return a solution of size $\alpha \cdot k$ or less (with probability at least $\frac{1}{2}$) if the optimum of the instance is at most k

Definition 3.1 (Parameterized Approximation). *Let Π be a hypergraph property and $\mathcal{G}$ be a closed set of hypergraphs. An algorithm $\mathcal{A}$ is a* randomized parameterized α-approximation algorithm *for $(\mathcal{G}, \Pi)$-*`Vertex Deletion` *if it takes a graph $G \in \mathcal{G}$ and an integer $k \geq 0$ as input, and returns a solution $S \in \texttt{SOL}_{\Pi}(G)$ which satisfies the following.*

- *If $\texttt{OPT}_{\mathcal{G},\Pi}(G) \leq k$, then $\Pr(|S| \leq \alpha \cdot k) \geq \frac{1}{2}$.*

Moreover, the running time of $\mathcal{A}$ is $f(k) \cdot n^{\mathcal{O}(1)}$ for some function f.

We note that the above definition of parameterized approximation algorithms captures the classic definition of exact parameterized algorithms ($\alpha = 1$) and polynomial time approximation algorithms ($f(k) = O(1)$) as special cases. Sampling with a Black-Box converts a randomized parameterized α-approximation algorithm $\mathcal{A}$ which runs in time $c^k \cdot n^{\mathcal{O}(1)}$ to a randomized parameterized β-approximation algorithm $\mathcal{B}$ which runs in time $d^k \cdot n^{\mathcal{O}(1)}$. The conversion relies on a simple problem dependent *sampling step*. In all our application the algorithm $\mathcal{A}$ is either the state-of-art exact parameterized algorithm (i.e., $\alpha = 1$), or the state-of-art polynomial time algorithm (i.e., $c = 1$) for the problem.

3.2 Sampling Steps

We exploit inherent properties of a specific $(\mathcal{G}, \Pi)$-`Vertex Deletion` problem to come up with basic sampling strategies. From a high-level perspective, a sampling step is a polynomial-time algorithm that takes as input a hypergraph $G \in \mathcal{G} \setminus \Pi$ and returns a vertex $v \in V(G)$, such that the size of the optimal solution of $G \setminus v$ decreases by 1, with a prescribed probability.

Definition 3.2 (Sampling Step). *Let $\mathcal{G}$ be a closed set of hypergraph, Π be a hypergraph property and $q \in (0,1)$. A* sampling step with success probability q *is a polynomial time randomized algorithm $\mathcal{R}$ that takes as input a hypergraph $G \in \mathcal{G} \setminus \Pi$ and returns a vertex $v \in V(G)$ such that*

$$\Pr\Big(\texttt{OPT}_{\Pi}(G \setminus v) \leq \texttt{OPT}_{\Pi}(G) - 1\Big) \geq q.$$

For example, consider VERTEX COVER, which is a $(\mathcal{G}, \Pi)$-`Vertex Deletion` problem where $\mathcal{G}$ is the set of all graphs and Π consists of all edgeless graphs. The following is a very simple sampling step for VERTEX COVER with success probability $\frac{1}{2}$: pick an arbitrary edge and return each of its endpoints with probability $\frac{1}{2}$. This algorithm clearly runs in polynomial time. Moreover, for each VERTEX COVER S of G and for each edge e, at least one of the endpoints of e belongs to S. Therefore, its a sampling step with success probability $q = \frac{1}{2}$ for VERTEX COVER. We provide sampling steps for the general case where Π is defined by a finite set of forbidden subgraphs, for FEEDBACK VERTEX SET and for PATHWIDTH ONE VERTEX DELETION.

On their own, sampling steps can be easily used to derive parameterized approximation algorithms. To obtain a β-approximation, one may use the sampling step $\beta \cdot k$ times, where each execution returns a vertex v which is removed from the graph and added to the solution S. After $\beta \cdot k$ steps, with some probability P, the set S of returned vertices is indeed a solution. Thus, by repeating this multiple sampling step for $\frac{1}{P}$ times, one gets a β-approximate solution with probability $\frac{1}{2}$. By carefully tracing the probability in the above argument, we show that $P \approx \left(\exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \middle\| q\right)\right)\right)^{-k}$, as demonstrated by the following lemma.[2]

Lemma 3.3. *Let $\mathcal{G}$ be a closed set of hypergraphs and Π be an hypergraph property such that there is a sampling step with success probability q for $(\mathcal{G}, \Pi)$-`Del`. Then for every $1 \leq \beta \leq \frac{1}{q}$ there is a randomized parameterized β-approximation for $(\mathcal{G}, \Pi)$-`Del` which runs in time $d^k \cdot n^{\mathcal{O}(1)}$ where $d = \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \middle\| q\right)\right)$.*

The proof of Lemma 3.3 can be found in Section 5. Observe that $\exp\left(\frac{1}{q} \cdot \mathcal{D}\left(\frac{1}{\left(\frac{1}{q}\right)} \middle\| q\right)\right) = 1$ for every $q \in (0, 1)$. Thus, Lemma 3.3 provides a polynomial time $\frac{1}{q}$-approximation algorithm for $(\mathcal{G}, \Pi)$-`Del`. This justifies the restriction of the lemma to $\beta \leq \frac{1}{q}$.

While Lemma 3.3 provides a parameterized β-approximation algorithm for essentially any β, the resulting algorithms are often clearly far from optimal. For example, for FEEDBACK VERTEX SET (`FVS`) we provide a sampling step with success probability $\frac{1}{4}$. Thus, by Lemma 3.3 we get a parameterized 1.1-approximation for `FVS` which runs in time $\approx 2.944^k \cdot n^{\mathcal{O}(1)}$. However, `FVS` has an *exact* parameterized algorithm which runs in time $2.7^k \cdot n^{\mathcal{O}(1)}$. That is, the running time of the approximation algorithm is slower than that of the exact algorithm. Our goal is to combine the sampling step with the exact algorithm to attain improved running time in such cases.

3.3 Sampling with a Black-Box

Our main theorem states the following: a sampling step with success probability q, together with a parameterized α-approximation algorithm $\mathcal{A}$ which runs in time $c^k \cdot n^{\mathcal{O}(1)}$, can be used to obtain a β-approximation algorithm $\mathcal{B}$. By Lemma 3.3, the sampling step can be used to obtain a randomized parameterized α-approximation which runs in time $\left(\exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\|\, q\right)\right)\right)^k \cdot n^{\mathcal{O}(1)}$. Our assumption is that $\mathcal{A}$ is at least as fast as the algorithm provided by Lemma 3.3, hence we only consider the case in which $c \leq \exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\|\, q\right)\right)$.

We use the following functions to express the running time of $\mathcal{B}$. Define two function $\delta^*_{\text{left}}(\alpha, c, q)$ and $\delta^*_{\text{right}}(\alpha, c, q)$ as the unique numbers $\delta^*_{\text{left}}(\alpha, c, q) \in (1, \alpha]$ and $\delta^*_{\text{right}}(\alpha, c, q) \in [\alpha, \infty)$ which satisfy

$$\mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{1}{\delta^*_{\text{left}}(\alpha, c, q)}\right) = \mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{1}{\delta^*_{\text{right}}(\alpha, c, q)}\right) = \mathcal{D}\left(\frac{1}{\alpha} \middle\| q\right) - \frac{\ln(c)}{\alpha}. \tag{2}$$

[2]Recall $\mathcal{D}(\cdot \,\|\, \cdot)$ stands for the Kullback-Leibler divergence which has been defined in Section 2.

We write $\delta^*_{\text{left}} = \delta^*_{\text{left}}(\alpha, c, q)$ and $\delta^*_{\text{right}} = \delta^*_{\text{right}}(\alpha, c, q)$ if the values of α, c and q are clear from the context. The following lemma provides conditions which guarantee that δ^*_{left} and δ^*_{right} are well defined in certain domains.

Lemma 3.4. *For every $c \geq 1$, $0 < q < 1$ and $\alpha \geq 1$ such that $c \leq \exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, q\right)\right)$, the value of $\delta^*_{\text{right}}(\alpha, c, q)$ is well defined. Furthermore, if $\alpha > 1$, then $\delta^*_{\text{left}}(\alpha, c, q)$ is also well defined.*

The proof of Lemma 3.4 can be found in Section 6. We note that $\mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, \frac{1}{x}\right)$ is monotone in the domains $x \in (1, \alpha]$ and $x \in [\alpha, \infty)$. Hence the values of $\delta^*_{\text{left}}(\alpha, c, q)$ and $\delta^*_{\text{right}}(\alpha, c, q)$ can be easily evaluated to arbitrary precision using a simple binary search.

We use δ^*_{left} and δ^*_{right} to express the running time of $\mathcal{B}$. For every $\beta, \alpha \geq 1$, $0 < q \leq 1$ and $c \geq 1$ such that $c \leq \exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, q\right)\right)$, we define

$$\texttt{convert}\,(\alpha, \beta, c, q) := \begin{cases} c \cdot \exp\left(\frac{\delta^*_{\text{right}} \cdot \mathcal{D}\left(\frac{1}{\delta^*_{\text{right}}} \,\middle\|\, q\right) - \ln(c)}{\delta^*_{\text{right}} - \alpha} \cdot (\beta - \alpha)\right) & \text{if } \delta^*_{\text{right}}(\alpha, c, q) > \beta \geq \alpha \\ c \cdot \exp\left(\frac{\delta^*_{\text{left}} \cdot \mathcal{D}\left(\frac{1}{\delta^*_{\text{left}}} \,\middle\|\, q\right) - \ln(c)}{\delta^*_{\text{left}} - \alpha} \cdot (\beta - \alpha)\right) & \text{if } \delta^*_{\text{left}}(\alpha, c, q) < \beta \leq \alpha \\ \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \,\middle\|\, q\right)\right) & \text{otherwise} \end{cases} \tag{3}$$

As we can compute δ^*_{left} and δ^*_{right}, it follows we can also compute $\texttt{convert}\,(\alpha, \beta, c, q)$ to arbitrary precision.

Recall that Lemma 3.3 provides a polynomial time $\frac{1}{q}$-approximation algorithm for $(\mathcal{G}, \Pi)$-`Del`. Consequently, we restrict our consideration to α and β values that are less than or equal to $\frac{1}{q}$. Our main technical result is the following.

Theorem 3.5 (Sampling with a Black-Box)**.** *Let $\mathcal{G}$ be a closed set of hypergraphs and Π be a hypergraph property. Assume the following:*

- *There is a sampling step with success probability $q \in (0, 1)$ for $(\mathcal{G}, \Pi)$-`Del`.*
- *There is a randomized parameterized α-approximation algorithm for $(\mathcal{G}, \Pi)$-`Del` which runs in time $c^k \cdot n^{\mathcal{O}(1)}$ for some $c \geq 1$, $1 \leq \alpha \leq \frac{1}{q}$ and $c \leq \exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, q\right)\right)$.*

Then, for every $1 \leq \beta \leq \frac{1}{q}$, there is a randomized parameterized β-approximation for $(\mathcal{G}, \Pi)$-`Del` which runs in time $\left(\texttt{convert}\,(\alpha, \beta, c, q)\right)^k \cdot n^{\mathcal{O}(1)}$.

The proof of Theorem 3.5 is given in Section 5. For example, by Theorem 3.5, we can use the sampling step with success probability $\frac{1}{4}$ for `FVS`, together with the exact parameterized algorithm for the problem which runs in time $2.7^k \cdot n^{\mathcal{O}(1)}$ [33], to get a randomized parameterized 1.1-approximation algorithm with running time $2.49^k \cdot n^{\mathcal{O}(1)}$. The algorithm achieves a better running time than that of the exact parameterized algorithm, as well as the running time which can be attain solely by the sampling step, i.e. $2.944^k \cdot n^{\mathcal{O}(1)}$ (Lemma 3.3). We note that this running time is also superior to the running time of $2.62^k \cdot n^{\mathcal{O}(1)}$ for the same approximation ratio given in [28].

The running time of the algorithm generated by Theorem 3.5 (i.e., the value of $\texttt{convert}\,(\alpha, \beta, c, q)$) can always be computed efficiently, though this computation requires a binary search for the evaluation of δ^*_{right} and δ^*_{left}. For the special cases of $\alpha = 1$ and well as ($\alpha = 2$ and $c = 1$) we provide a closed form expression for $\texttt{convert}\,(\alpha, \beta, c, q)$.

Theorem 3.6 (simple formula for $\alpha = 1$)**.** *For every $0 < q < 1$, $1 \leq \beta \leq \frac{1}{q}$ and $c \geq 1$ such that $c \leq \exp\left(1 \cdot \mathcal{D}\left(\frac{1}{1} \,\middle\|\, q\right)\right) = \frac{1}{q}$ it holds that*

$$\texttt{convert}\,(1, \beta, c, q) = \begin{cases} c \cdot \left(\frac{1-c\cdot q}{1-q}\right)^{\beta - 1} & \text{if } 1 \leq \beta < \frac{1}{q \cdot c} \\ \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \,\middle\|\, q\right)\right) & \text{if } \frac{1}{q\cdot c} \leq \beta \leq \frac{1}{q} \end{cases}$$

Theorem 3.7 (simple formula for $\alpha = 2$ and $c = 1$)**.** *Let $0 < q \leq \frac{1}{2}$ and $1 \leq \beta \leq 2$. Then it holds that*

$$\texttt{convert}\,(2, \beta, 1, q) = \begin{cases} \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \,\middle\|\, q\right)\right) & \text{if } 1 \leq \beta \leq \frac{1}{1-q} \\ \left(\frac{q}{1-q}\right)^{\beta - 2} & \text{if } \frac{1}{1-q} < \beta \leq 2 \end{cases}$$

Both Theorems 3.6 and 3.7 follow from a closed form formula which we can attain for δ^*_{right} or δ^*_{left} for the specific values of α and c considered in the theorems. The proof of Theorems 3.6 and 3.7 is given in Section 6.

3.4 Comparison to Fidelity Preserving Transformations

We compare our technique, Sampling with a Black-Box, with the technique of [23] for $(\mathcal{G}, \Pi)$-`Vertex Deletion` problems in which Π is defined by a finite set of forbidden vertex induced hypergraphs.

Definition 3.8. *Let $\Omega = \{F_1, \ldots, F_\ell\}$ be a finite set of hypergraphs for $\ell > 0$. Then Π^Ω is the hypergraph property where a hypergraph G belongs to Π^Ω if and only if there is no vertex induced subhypergraph X of G such that X is isomorphic to F_i, for some $1 \leq i \leq \ell$.*

We note that Π^Ω is always hereditary and polynomial-time decidable. We also note the the family of graphs properties defined by a finite set of forbidden hypergraph suffices to define many fundamental graph problems such as Vertex Cover, d-Hitting Set, ℓ-Path Vertex Cover and Directed Feedback Vertex Set on Tournaments.

For a set of hypergraphs $\Omega = \{F_1, \ldots, F_\ell\}$, define $\eta(\Omega) := \max_{1 \leq i \leq \ell} |V(F_i)|$, the maximal number of vertices of a hypergraph in Ω. The following result has been (implicitly) given in [23].

Lemma 3.9 ([23])**.** *Let Ω be a finite set of hypergraphs and let $\mathcal{G}$ be a closed set of hypergraphs. Furthermore, assume there is an randomized exact parameterized $c^k \cdot n^{\mathcal{O}(1)}$ algorithms for $(\mathcal{G}, \Pi^\Omega)$-`Vertex Deletion`. Then for every $1 \leq \beta \leq \eta(\Omega)$ there is a randomized parameterized β-approximation algorithm for $(\mathcal{G}, \Pi^\Omega)$-`Vertex Deletion` which runs in time*

$$c^{\frac{\eta(\Omega) - \beta}{\eta(\Omega) - 1} \cdot k} \cdot n^{\mathcal{O}(1)}.$$

There is a simple and generic way to design a sampling step for Π^Ω. The sampling step finds a set of vertices $S \subseteq V(G)$ of the input graph such that $G[V]$ is isomorphic for a graph in Ω, and returns a vertex from S uniformly at random. This leads to the following lemma.

Lemma 3.10. *Let Ω be a finite set of hypergraphs and let $\mathcal{G}$ be a closed set of hypergraphs. There is a sampling step for $(\mathcal{G}, \Pi^\Omega)$-`Vertex Deletion` with success probability $\frac{1}{\eta(\Omega)}$.*

A formal proof for Lemma 3.10 is given in Section 7. Together with Theorem 3.5 the lemma implies the following.

Corollary 3.11. *Let Ω be a finite set of hypergraphs and $\mathcal{G}$ be a closed set of hypergraphs. Furthermore, assume there is a randomized exact parameterized $c^k \cdot n^{\mathcal{O}(1)}$ algorithms for $(\mathcal{G}, \Pi^{\Omega})$-*`Vertex Deletion`*. Then for every $1 \leq \beta \leq \eta(\Omega)$ there is a randomized parameterized β-approximation algorithm for $(\mathcal{G}, \Pi^{\Omega})$-*`Vertex Deletion` *which runs in time*

$$\left(\texttt{convert}\left(1, \beta, c, \frac{1}{\eta(\Omega)}\right)\right)^k \cdot n^{\mathcal{O}(1)}.$$

Observe that Lemma 3.9 and Corollary 3.11 only differ in the resulting running time. The following lemma implies that for every $1 < \beta < \eta(\Omega)$ the running time of Corollary 3.11, the running time of Sampling with a Black-Box, is always strictly better than the running time of Lemma 3.9, the result of [23].

Lemma 3.12. *For every $\eta \in \mathbb{N}$ such that $\eta \geq 2$, $1 < c < \eta$ and $1 < \beta < \eta$ it holds that*

$$\texttt{convert}\left(1, \beta, c, \frac{1}{\eta}\right) < c^{\frac{\eta - \beta}{\eta - 1}}.$$

The proof of Lemma 3.12 is given in Section 9.

4 Applications

In this section we will describe some problems to which our results can be applied. For each problem, we utilize sampling steps to obtain parameterized approximation algorithms. We also compare the running time of our algorithm with a benchmark whenever applicable.

4.1 Feedback Vertex Set

Recall that given a graph G and integer k, the Feedback Vertex Set problem asks whether there exists a set $S \subseteq V(G)$ of size at most k such that $G \setminus S$ is acyclic. Feedback Vertex Set can also be described as a $(\mathcal{G}, \Pi)$-`Vertex Deletion` problem, where $\mathcal{G}$ is the set of all graphs and Π is the set of graphs that have no cycles. First, we demonstrate that there exists a sampling step for Feedback Vertex Set.

Lemma 4.1. Feedback Vertex Set *has a sampling step with success probability $\frac{1}{4}$.*

The proof of Lemma 4.1 can be found in Section 7.1. The sampling step presented in Section 7.1 begins by removing vertices of degree at most 1. Then, a vertex is sampled from the remaining vertices, where the sampling probability for each vertex is proportional to its degree in the remaining graph.

In [33], the authors present an FPT algorithm which runs in time $2.7^k \cdot n^{\mathcal{O}(1)}$ (i.e., in the terminology of this paper, $\alpha = 1, c = 2.7$). Moreover, Feedback Vertex Set also has a 2-approximation algorithm that runs in polynomial time (i.e., $\alpha = 2$, $c = 1$) [2]. In the following, we demonstrate how the sampling step, together with the existing algorithms, is used to develop a new approximation algorithm with a better running time.

Theorem 4.2. *For each $1 \leq \beta \leq 2$,* Feedback Vertex Set *has a β-approximation algorithm which runs in time $d^k \cdot n^{\mathcal{O}(1)}$ where*

$$d = \begin{cases} 2.7 \cdot \left(\frac{1.3}{3}\right)^{\beta - 1} & \text{if } 1 \leq \beta < 1.402 \\ \left(\frac{1}{3}\right)^{\beta - 2} & \text{if } 1.402 \leq \beta \leq 2 \end{cases}. \tag{4}$$

Proof. TOPROVE 2 □

In [28], the authors present a β-approximation algorithm for each $1 < \beta \leq 2$. It can be visually (see Fig. 1) and numerically (see Table 1) verified that our algorithm demonstrates a strictly better running-time .

β	Our Algorithm (??)	[28]
1.1	2.483	2.620
1.2	2.284	2.467
1.3	2.101	2.160
1.4	1.932	1.942
1.5	1.732	1.778
1.6	1.552	1.649
1.7	1.390	1.56
1.8	1.246	1.319
1.9	1.116	1.149

Table 1: Comparison of the base of exponents of different algorithms for FVS. For each row with a β value b, a value d in the second or third column implies a b-approximation algorithm with running time $d^k \cdot n^{\mathcal{O}(1)}$.

4.2 Pathwidth One Vertex Deletion

Given a graph G and integer k, the Pathwidth One Vertex Deletion (POVD) problem ask whether there exists a set $S \subseteq V(G)$ of size at most k such that $G \setminus S$ has pathwidth at most 1. Initially, we demonstrate that there exists a sampling step for POVD.

Lemma 4.3. Pathwidth One Vertex Deletion *has a sampling step with probability* $\frac{1}{7}$.

The proof of Lemma 4.3 can be found in Section 7.2. The sampling step for POVD is very similar to that for FVS, with a slight modification. Similar to FVS, POVD can be described by a set of forbidden subgraphs. Moreover, the set of forbidden subgraphs for POVD includes one additional graph with 7 vertices. Therefore POVD has a sampling step with probability $\frac{1}{7}$, instead of $\frac{1}{4}$ as in the case of FVS.

To the best of our knowledge, parameterized approximation algorithms have not been studied for POVD. However, there exists an FPT algorithm for POVD with running time $3.888^k \cdot n^{\mathcal{O}(1)}$ ($\alpha = 1, c = 3.888$) [44]. Next, we demonstrate how combining the aforementioned sampling step with a parameterized algorithm yields a new approximation algorithm. Refer to Fig. 3 and Table 2 for the corresponding running time.

Theorem 4.4. *For each* $1 \leq \beta \leq 7$, Pathwidth One Vertex Deletion *has a* β*-approximation algorithm which runs in time* $d^k \cdot n^{\mathcal{O}(1)}$ *where*

$$d = \begin{cases} 3.888 \cdot (0.519)^{\beta-1} & \text{if } 1 \leq \beta \leq 1.8 \\ \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \middle\| \frac{1}{7}\right)\right) & \text{if } 1.8 < \beta < 7. \end{cases} \tag{5}$$

Proof. TOPROVE 3 □

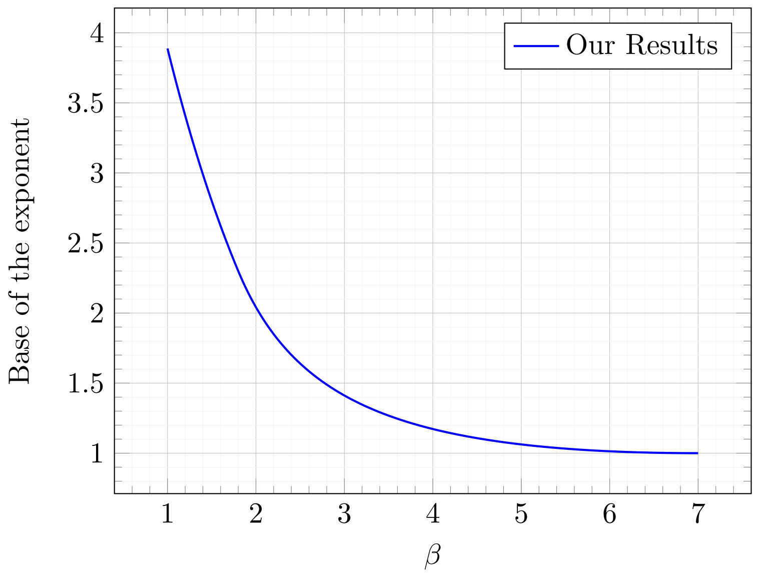

Figure 3: A plot of the running time of our algorithm for Pathwidth One Vertex Deletion. The x-axis corresponds to the approximation ratio, while the y-axis corresponds to the base of the exponent in the running time. A point (β, d) in the plot describes a running time of the form $d^k \cdot n^{\mathcal{O}(1)}$ for a β-approximation.

4.3 $(\mathcal{G}, \Pi)$-Vertex Deletion for a finite set of forbidden sub-hypergraphs

There are many problems that can be described as $(\mathcal{G}, \Pi)$-`Vertex Deletion` problems in which Π is defined by a finite set Ω of forbidden vertex induced hypergraphs. For each of those problems, by Lemma 3.10 there exists a sampling step with success probability $\frac{1}{\eta}$, where η is the maximum number of vertices of a hypergraph in Ω. In the following, we will demonstrate how we can obtain parameterized approximation algorithms for such problems. For the sake of presentation, we will focus on a specific problem called 3-Path Vertex Cover.

Given a graph G, a subset of vertices $S \subseteq V(G)$ is called an ℓ-path Vertex Cover if every path of length ℓ contains a vertex from S. The ℓ-Path Vertex Cover problem asks whether there exists an ℓ-path Vertex Cover of size at most k where k is the parameter [11]. ℓ-Path Vertex Cover can be described as a $(\mathcal{G}, \Pi)$-`Vertex Deletion` where $\mathcal{G}$ is the set of graphs and Π is the set of graphs with maximum path length at most $\ell - 1$. Alternatively, let F be a path with ℓ vertices where we define $\Omega := \{F\}$ and $\eta(\Omega) := \ell$. It holds that ℓ-Path Vertex Cover is equivalent to $(\mathcal{G}, \Pi^{\Omega})$-`Vertex Deletion`. Therefore, by Lemma 3.10, there is a sampling step for ℓ-Path Vertex Cover with success probability $\frac{1}{\ell}$.

In the following, we will consider $\ell = 3$, i.e. the problem 3-Path Vertex Cover.

There exists an FPT algorithm for t-Path Vertex Cover hat runs in time $1.708^k \cdot n^{\mathcal{O}(1)}$ $(\alpha = 1, c = 1.708)$ [14]. Moreover, there is also a 2-approximation algorithm that runs in polynomial time. $(\alpha = 2, c = 1)$ [45].

Theorem 4.5. *For each $1 < \beta < 2$, 3-Path Vertex Cover has a β-approximation algorithm which runs in time $d^k \cdot n^{\mathcal{O}(1)}$ where*

$$d = \begin{cases} 1.708 \cdot (0.644)^{\beta-1} & \text{if } 1 \le \beta < 1.6143 \\ (0.5)^{\beta-2} & \text{if } 1.6143 \le \beta \le 2 \end{cases} \tag{6}$$

Proof. TOPROVE 4 □

β	Value	β	Value	β	Value	β	Value	β	Value	β	Value
1.1	3.6412	2.1	1.9391	3.1	1.3776	4.1	1.1573	5.1	1.0553	6.1	1.0107
1.2	3.4100	2.2	1.8498	3.2	1.3466	4.2	1.1433	5.2	1.0488	6.2	1.0083
1.3	3.1936	2.3	1.7713	3.3	1.3181	4.3	1.1303	5.3	1.0428	6.3	1.0063
1.4	2.9908	2.4	1.7018	3.4	1.2920	4.4	1.1183	5.4	1.0374	6.4	1.0046
1.5	2.8010	2.5	1.6399	3.5	1.2679	4.5	1.1071	5.5	1.0323	6.5	1.0031
1.6	2.6232	2.6	1.5844	3.6	1.2457	4.6	1.0968	5.6	1.0277	6.6	1.0020
1.7	2.4567	2.7	1.5346	3.7	1.2252	4.7	1.0871	5.7	1.0236	6.7	1.0011
1.8	2.3007	2.8	1.4895	3.8	1.2062	4.8	1.0782	5.8	1.0198	6.8	1.0005
1.9	2.1604	2.9	1.4487	3.9	1.1886	4.9	1.0700	5.9	1.0164	6.9	1.0001
2.0	2.0417	3.0	1.4115	4.0	1.1724	5.0	1.0624	6.0	1.0134		

Table 2: The table displays the base of exponents for our algorithm designed for Pathwidth One Vertex Deletion. Each pair (b, d), listed in the same row and consecutive β and value columns, represents a β-approximation algorithm with a running time $d^k \cdot n^{\mathcal{O}(1)}$.

In [23], for each $1 \leq \beta \leq 2$, the authors present a β-approximation algorithm for 3-Path Vertex Cover with running time $1.708^{\frac{3-\beta}{2} \cdot k} \cdot n^{\mathcal{O}(1)}$. As can be visually (see Fig. 2) or numerically (see Table 3) verified, our algorithm has a strictly better running time for all values of $1 < \beta \leq 2$.

β	Our Algorithm (6)	[23]
1.1	1.6345	1.6628
1.2	1.5641	1.6189
1.3	1.4968	1.5762
1.4	1.4323	1.5345
1.5	1.3707	1.4940
1.6	1.3117	1.4545
1.7	1.2311	1.416
1.8	1.1487	1.3787
1.9	1.0718	1.3423

Table 3: Comparison of the base of exponents of different algorithms for 3-Path Vertex Cover. For each row with a β value b, a value d in the second or third column implies a b-approximation algorithm with running time $d^k \cdot n^{\mathcal{O}(1)}$.

5 Sampling with a Black Box

In this section we present our main technique, Sampling with a Black Box, and prove Theorem 3.5. The technique is designed using three main components, enabling a modular analysis of each part. We use the notion of (δ, p, r, T)-`procedure` to abstract the outcome of iteratively executing a sampling step. We use this abstract notion in `RandAndExtend` which combines the (δ, p, r, T)-procedure together with the black box parameterized α-approximation algorithm. On its own, `RandAndExtend` only attains a β-approximate solution with a low probability. Our main algorithm, `SamplingWithABlackBox`, executes `RandAndExtend` multiple times to get a β-approximate solution with a constant probability. The defined algorithm depends on a parameter δ, for which we find the optimal value.

We start with the formal definition of a (δ, p, r, T)`-procedure`. As already mentioned, a procedure serves as an abstraction of iterative use of a sampling step. It returns a vertex set $S \subseteq V(G)$ with certain properties related to the size of S and the value of $\texttt{OPT}(G \setminus S)$.

Definition 5.1. *Let Π be a hypergraph property and $\mathcal{G}$ be a closed set of hypergraphs. For all $\delta \geq 1$, $r \geq 0$, $0 < p \leq 1$ and $T \geq 0$, a (δ, p, r, T)`-procedure` for $(\mathcal{G}, \Pi)$ is a polynomial time randomized algorithm that takes as input a hypergraph $G \in \mathcal{G}$, an integer $t \geq 0$ and returns a set $S \subseteq V(G)$ with the following properties:*

P1 *It holds that $|S| \leq \delta \cdot t$.*

P2 *Suppose that $\texttt{OPT}_{\mathcal{G},\Pi}(G) \leq k$ for some $k \geq 0$. If $t \geq T$, then with probability at least $\frac{p^t}{(t+1)^r}$ it holds that $G \setminus S$ has a solution of size at most $\max(0, k-t)$, i.e.*

$$\Pr\Big(\texttt{OPT}_{\mathcal{G},\Pi}(G \setminus S) \leq \max(0, k-t)\Big) \geq \frac{p^t}{(t+1)^r}.$$

Additionally, we use the notation (δ, p)`-procedure` to refer to a (δ, p, r, T)`-procedure` for some constants $r, T \geq 0$.

Observe that if there is a (δ, p)`-procedure` for $(\mathcal{G}, \Pi)$`-Vertex Deletion`, then we can make use of it to obtain a δ approximation as follows. Suppose G has a solution of size k. Then, by setting $t = k$ in Property **P2**, it holds that with probability at least $\frac{p^k}{(k+1)^r}$, $G \setminus S$ has a solution of size 0, i.e. $G \setminus S \in \Pi$. By our assumption, we can check in polynomial time whether a graph belongs to the property Π. By Property **P1** it holds that $|S| \leq \delta \cdot k$. Additionally, we can repeat this algorithm $p^{-k} \cdot n^{\mathcal{O}(1)}$ times to obtain a δ-approximate solution constant probability. We summarize these insights in the following observation.

Observation 5.2. *If there is a (δ, p)`-procedure` for $(\mathcal{G}, \Pi)$`-Vertex Deletion`, then there is parameterized δ-approximation algorithm for $(\mathcal{G}, \Pi)$`-Vertex Deletion` with running time $(1/p)^k \cdot n^{\mathcal{O}(1)}$.*

In the remainder of Section 5, we fix the values of $0 < q \leq 1$, $1 \leq \alpha \leq \frac{1}{q}$, $1 \leq \beta \leq \frac{1}{q}$ and $1 \leq c \leq \exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, q\right)\right)$ to specific numbers, unless specified explicitly. For notational simplicity, we omit α, β, and c from the subscript of functions dependent on these variables. Moreover, let Π be a fixed polynomial-time decidable hypergraph property, $\mathcal{G}$ be a closed set of hypergraphs and $\mathcal{A}$ be an α-approximation algorithm for the $(\mathcal{G}, \Pi)$`-Vertex Deletion` problem with running time $c^k \cdot n^{\mathcal{O}(1)}$. Let us also define the set of values δ can take, given α and β.

Definition 5.3. *For $\alpha, \beta \geq 1$ such that $\alpha \neq \beta$, we define the set* `interval`(α, β) *as*

$$\texttt{interval}(\alpha,\beta) := \begin{cases} [\beta, \infty) & \textit{if } \beta > \alpha \\ [1, \beta] & \textit{if } \beta < \alpha. \end{cases}$$

The next component in our technique is `RandAndExtend`, given in Algorithm 1. The algorithm is configured with a (δ, p)`-procedure` $\mathcal{P}_{\delta,p}$. Given an hypergraph G and an integer t in the input, the algorithm invokes the procedure $\mathcal{P}_{\delta,p}$ which returns a random set $S \subseteq V(G)$ of size at most $\delta \cdot t$, and then runs the parameterized α-approximation algorithm $\mathcal{A}$ on the remaining hypergraph $G \setminus S$. The idea is to hope that $G \setminus S$ has a solution of size at most $\frac{\beta \cdot k - \delta \cdot t}{\alpha}$. Note that the parameter for the α-approximation algorithm is also $\frac{\beta \cdot k - \delta \cdot t}{\alpha}$, which ensures that the approximation algorithm

Algorithm 1 `RandAndExtend`

Configuration: $0 < p \leq 1$, $\delta \in \texttt{interval}(\alpha, \beta)$ and a (δ, p, r, T)-`procedure` $\mathcal{P}_{\delta,p}$ for $(\mathcal{G}, \Pi)$.

Input: Hypergraph $G \in \mathcal{G}$, integers $0 \leq k \leq |V(G)|$ and $T \leq t \leq \frac{\beta}{\delta} \cdot k$

```
1: $S = \mathcal{P}_{\delta,p}(G, t)$
2: $Y = \mathcal{A}\left(G \setminus S, \frac{\beta \cdot k - \delta \cdot t}{\alpha}\right)$
3: Return $S \cup Y$
```

returns a set of size at most $\beta \cdot k - \delta \cdot t$, with high probability. In the event that this holds, by adding S to the returned set, we obtain a solution with a size of at most $\beta \cdot k$.

Our main algorithm, given in Algorithm 2, begins by selecting a value for t^*. This value ensures the following: if the set S in Algorithm 1 contains at least t^* many elements from a solution, then $G \setminus S$ has a solution of size at most $\frac{\beta \cdot k - \delta \cdot t^*}{\alpha}$. Note that this further implies that the set returned by Algorithm 1 has size at most $\beta \cdot k$. Furthermore, Algorithm 2 utilizes Algorithm 1 and executes it multiple times to ensure a β-approximate solution is attained with a constant probability.

Algorithm 2 `SamplingWithABlackBox`

Configuration: $0 < p \leq 1$, $\delta \in \texttt{interval}(\alpha, \beta)$ and a (δ, p, r, T)-`procedure` $\mathcal{P}_{\delta,p}$ for $(\mathcal{G}, \Pi)$.

Input: Hypergraph $G \in \mathcal{G}$, integer $0 \leq k \leq |V(G)|$

```
1: $t^* := \left\lceil \frac{\beta - \alpha}{\delta - \alpha} \cdot k \right\rceil$ if $\beta < \alpha$, and $t^* := \left\lfloor \frac{\beta - \alpha}{\delta - \alpha} \cdot k \right\rfloor$ if $\beta > \alpha$
2: if $t^* < T$ then
3:     Return $W$ if and only if there exists $W \subseteq V(G)$ of size at most $k$ such that $W \in \texttt{SOL}_\Pi(G)$
4: else
5:     $\mathcal{S} = \emptyset$
6:     for $2 \cdot p^{-t^*} \cdot (t^* + 1)^r$ times do
7:         $\mathcal{S} = \mathcal{S} \cup \left\{ \texttt{RandAndExtend}(G, k, t^*) \right\}$
8:     Return a minimum sized set in $\mathcal{S}$
```

Lemma 5.4. *Let $0 < p \leq 1$, $\delta \in \texttt{interval}(\alpha, \beta)$ and a (δ, p, r, T)-`procedure` $\mathcal{P}_{\delta,p}$ for $(\mathcal{G}, \Pi)$. Then Algorithm 2 is a randomized parameterized β-approximation algorithm for $(\mathcal{G}, \Pi)$-`Del` with running time*

$$f(\delta, p)^k \cdot n^{\mathcal{O}(1)}$$

where $f(\delta, p)$ is given by

$$f(\delta, p) := \exp\left(\frac{(\delta - \beta) \cdot \ln(c) + (\beta - \alpha) \cdot \ln\left(\frac{1}{p}\right)}{\delta - \alpha}\right).$$

The proof of Lemma 5.4 can be found in Section 5.1.

Algorithm 2 relies on the existence of a (δ, p)-`procedure`, however, insofar we did not show how to design one. We generate a (δ, p)-`procedure` from a sampling step (Definition 3.2) via a simple algorithm which iteratively invokes the sampling step. The pseudo-code of the algorithm is given in Algorithm 3.

Define

$$\phi(\delta, q) := \exp\left(-\delta \cdot \mathcal{D}\left(\frac{1}{\delta} \middle\| q\right)\right). \tag{7}$$

The following lemma state that Algorithm 3 is indeed a (δ, p)-`procedure` for $p = \phi(\delta, q)$.

Algorithm 3 MultiSample

Configuration: A number $\delta \geq 1$ and a sampling step $\mathcal{R}$ for $(\mathcal{G}, \Pi)$-`Del` with success probability q for some $0 < q \leq 1$.

Input: Hypergraph $G \in \mathcal{G}$, integer $t \geq 0$

```
1: $S \leftarrow \emptyset$
2: while $|S| < \delta \cdot t$ and $G \notin \Pi$ do
3:     $v = \mathcal{R}(G)$
4:     $G = G \setminus \{v\}$
5:     $S = S \cup \{v\}$
6: return $S$
```

Lemma 5.5. *Let $0 < q \leq 1$ and $\mathcal{R}$ be a sampling step for $(\mathcal{G}, \Pi)$-`Vertex Deletion` with success probability q. Then, for any $1 \leq \delta \leq \frac{1}{q}$, Algorithm 3 is a $(\delta, \phi(\delta, q))$-`procedure` for $(\mathcal{G}, \Pi)$-`Del`.*

The proof of Lemma 5.5 can be found in Section 5.2. Lemma 5.5 implies that for $\delta = \frac{1}{q}$, there exists a $\left(\frac{1}{q}, 1\right)$-`procedure` for $(\mathcal{G}, \Pi)$-`Vertex Deletion`. Observe that this serves as a $(\delta, 1)$-`procedure` for $\delta > \frac{1}{q}$. Therefore, we make the following observation.

Observation 5.6. *Let $0 < q \leq 1$ and $\mathcal{R}$ be a sampling step for $(\mathcal{G}, \Pi)$-`Vertex Deletion` with success probability q. Then, for any $\delta > \frac{1}{q}$, there is a $(\delta, 1)$-`procedure` for $(\mathcal{G}, \Pi)$-`Vertex Deletion`.*

Furthermore, we can now prove Lemma 3.3 using Lemma 5.5 together with Observation 5.2.

Proof. TOPROVE 5 ☐

Using the procedure from Lemma 5.5 together with Lemma 5.4 (Algorithm 2) we get the following results.

Lemma 5.7. *Suppose that:*

- *There is a sampling step with success probability $q \in (0, 1)$ for $(\mathcal{G}, \Pi)$-`Del`.*
- *There is a randomized parameterized α-approximation algorithm for $(\mathcal{G}, \Pi)$-`Del` which runs in time $c^k \cdot n^{\mathcal{O}(1)}$.*

Then, there is a randomized parameterized β-approximation algorithm for $(\mathcal{G}, \Pi)$-`Del` with running time

$$\left(\min_{\delta \in \texttt{interval}(\alpha, \beta) \cap [1, \frac{1}{q}]} \tilde{f}(\delta, q)\right)^k \cdot n^{\mathcal{O}(1)}$$

where $\tilde{f}(\delta, q)$ is defined as

$$\tilde{f}(\delta, q) := f(\delta, \phi(\delta, q)) = \exp\left(\frac{(\delta - \beta) \cdot \ln(c) + (\beta - \alpha) \cdot \ln\left(\frac{1}{\phi(\delta, q)}\right)}{\delta - \alpha}\right).$$

Lemma 5.7 provides a β approximation algorithms whose running time is a solution for an optimization problem. The final step towards the proof of Theorem 3.5 is to solve this optimization problem.

Lemma 5.8. *It holds that*

$$\min_{\delta \in \mathtt{interval}(\alpha,\beta)\cap[1,\frac{1}{q}]} \tilde{f}\left(\delta, q\right) = \mathtt{convert}\left(\alpha, \beta, c, q\right).$$

The proofs of Lemmas 5.7 and 5.8 can be found in Section 5.3. We now have everything to proceed with the proof of Theorem 3.5.

Proof. TOPROVE 6 □

5.1 Converting Procedures to Approximation Algorithms

In this section we will prove Lemma 5.4. To accomplish this, we will examine some properties of Algorithm 1. First, we will show that Algorithm 1 indeed returns a solution $W \in \mathtt{SOL}_\Pi(G)$. Subsequently, we will establish that with a certain probability, the set returned has size at most $\beta \cdot k$. Equipped with these results, we will proceed with the proof of Lemma 5.4.

Lemma 5.9. *Let $G \in \mathcal{G}$ be a hypergraph, and $0 \leq k \leq n$, $T \leq t \leq \frac{\beta}{\delta} \cdot k$ be integers. Let W denote the set returned by* $\mathtt{RandAndExtend}(G, k, t)$*, then it holds that $W \in \mathtt{SOL}_\Pi(G)$.*

Proof. TOPROVE 7 □

Lemma 5.10. *Let t^* be as defined in Algorithm 2, then it holds that*

$$\max\left(0, \frac{\beta - \alpha}{\delta - \alpha} \cdot k - 1\right) \leq t^* \leq \frac{\beta - \alpha}{\delta - \alpha} \cdot k + 1.$$

Moreover, it also holds that $t^* \leq \frac{\beta}{\delta} \cdot k$.

Proof. TOPROVE 8 □

Lemma 5.11. *Let $G \in \mathcal{G}$ be a hypergraph, $0 \leq k \leq n$ be an integer and let t^* be as defined in Algorithm 2 such that $t^* \geq T$. Moreover, let Z be the set returned by* $\mathtt{RandAndExtend}(G, k, t^*)$*. If* $\mathtt{OPT}_{\mathcal{G},\Pi}(G) \leq k$*, then $|Z| \leq \beta \cdot k$ with probability at least $\frac{p^{t^*}}{2 \cdot (t^*+1)^r}$.*

Proof. TOPROVE 9 □

Now we are ready to prove Lemma 5.4.

Proof. TOPROVE 10 □

5.2 Converting Sampling Steps to Procedures

In this section, we will prove Lemma 5.5 by developing several auxiliary lemmas. Let $0 < q \leq 1$, $1 \leq \delta \leq \frac{1}{q}$ and $\mathcal{R}$ be a sampling step for $(\mathcal{G}, \Pi)$-$\mathtt{Vertex\ Deletion}$, with success probability q. Consider Algorithm 3 with these parameters. We will demonstrate that there exists integers r and T such that Algorithm 3 is a $(\delta, \phi\left(\delta, q\right), r, T)$-$\mathtt{procedure}$. To that end, we will need to show that given a hypergraph $G \in \mathcal{G}$ and $t \geq 0$ as input, Algorithm 3 runs in polynomial time and satisfies properties **P1** and **P2**, as defined in Definition 5.1.

Note that neither the running time of the algorithm nor property **P1** depend on the values of r and T. Therefore, irrespective of the values of r and T, we will show that Algorithm 3 runs in polynomial time and that property **P1** holds. Then, we will show that there exists r and T for which property **P2** holds, implying the truth of Lemma 5.5.

Lemma 5.12. *Algorithm 3 runs in polynomial time.*

Proof. TOPROVE 11 □

Lemma 5.13. *Algorithm 3 satisfies property **P1** in Definition 5.1.*

Proof. TOPROVE 12 □

Next, we demonstrate a simple feature of hereditary hypergraph properties. Intuitively, removing a vertex from a hypergraph does not increase the size of the optimal solution. The proof of Lemma 5.14 can be found in Appendix D.

Lemma 5.14. *Let Π be a hereditary hypergraph property and G be a hypergraph. For any $v \in V(G)$, it holds that*

$$0 \leq \mathtt{OPT}_\Pi(G) - \mathtt{OPT}_\Pi(G \setminus v) \leq 1.$$

Let $\xi_1, \ldots, \xi_n$ be i.i.d. binary random variables and $\nu \in (0,1]$ such that $\Pr(\xi_i = 1) \geq \nu$ for all $1 \leq i \leq n$. The following inequality can be shown using standard arguments

$$\Pr\left(\sum_{i=1}^{n} \xi_i \geq t\right) \geq \exp\left(-\frac{n}{t} \cdot \mathcal{D}\left(\frac{t}{n} \,\middle\|\, \nu\right)\right) \cdot n^{\mathcal{O}(1)}.$$

In the following lemma, we prove a similar statement in our setting where the i.i.d. assumption is dropped. Its proof can be found in Appendix B.

Lemma 5.15. *Let $\delta, t \geq 1$ be integers, $\nu \in (0,1]$ be a real number and $\xi_1, \ldots, \xi_{\lfloor \delta \cdot t \rfloor} \in \{0,1\}$ be random variables such that*

$$\Pr(\xi_j = 1 \,|\, \xi_1 = x_1, \ldots, \xi_{j-1} = x_{j-1}) \geq \nu$$

for all $1 \leq j \leq \lfloor \delta \cdot t \rfloor$ and $(x_1, \ldots, x_{j-1}) \in \{0,1\}^{j-1}$. Then, there exist integers r and T that depend on δ, such that for $t \geq T$ it holds that

$$\Pr\left(\sum_{j=1}^{\lfloor \delta \cdot t \rfloor} \xi_j \geq t\right) \geq (\delta \cdot t + 1)^{-r} \cdot \exp\left(-\delta \cdot \mathcal{D}\left(\frac{1}{\delta} \,\middle\|\, \nu\right)\right)^t.$$

Given a hypergraph $G \in \mathcal{G}$ and $t \geq 0$ as input, let $\ell \leq \lfloor \delta \cdot t \rfloor$ be the number of iterations of the while loop in Algorithm 3. Let $G_0 \coloneqq G$ and for $1 \leq i \leq \ell$, let G_i denote the hypergraph G at the end of the i'th iteration. Similarly, let v_i denote the vertex v at the end of the i'th iteration, i.e. $v_i = \mathcal{R}(G_{i-1})$. For $\ell + 1 \leq i \leq \lfloor \delta \cdot t \rfloor$, we let $G_i \coloneqq G_{i-1}$ and $v_i \coloneqq v_{i-1}$. Furthermore, we define the random variables $Z_0 \coloneqq 0$ and

$$Z_i = \begin{cases} \mathtt{OPT}(G_{i-1}) - \mathtt{OPT}(G_i) & \text{if } 1 \leq i \leq \ell \\ 1 & \text{if } \ell < i \leq \lfloor \delta \cdot t \rfloor \end{cases}$$

for $1 \leq i \leq \lfloor \delta \cdot t \rfloor$. Intuitively, for $1 \leq i \leq \ell$, Z_i measures the decrease in the optimal solution size, from G_{i-1} to G_i. Note that, by definition, for $i \leq \ell$ it holds that $G_i = G_{i-1} \setminus \{v\}$ for some $v \in V(G_{i-1})$. Therefore, we have $Z_i \in \{0,1\}$ by Lemma 5.14.

Lemma 5.16. *For $1 \leq j \leq \lfloor \delta \cdot t \rfloor$ and $(x_1, \ldots, x_{j-1}) \in \{0,1\}^{j-1}$ it holds that*

$$\Pr\Big(Z_j = 1 \mid Z_1 = x_1, \ldots, Z_{j-1} = x_{j-1}\Big) \geq q.$$

Proof. TOPROVE 13 □

In the following lemma, we establish a lower bound on the probability that the graph returned by the algorithm, i.e. $G_{\lfloor \delta \cdot t \rfloor}$, has a solution of size at most $\max(0, k-t)$. This lower bound is equal to the probability that the sum of Z_i for i from 1 to $\lfloor \delta \cdot t \rfloor$ exceeds t.

Lemma 5.17. *It holds that*

$$\Pr\Big(G_{\lfloor \delta\cdot t\rfloor} \textit{ has a solution of size at most } \max(0,k-t)\Big) \geq \Pr\left(\sum_{i=1}^{\lfloor \delta\cdot t\rfloor} Z_i \geq t\right).$$

Proof. TOPROVE 14 □

Lemma 5.18. *There exists $r, T \geq 0$ such that Algorithm 3 satisfies property **P2** in Definition 5.1.*

Proof. TOPROVE 15 □

Finally, the proof of Lemma 5.5 simply follows from Lemmas 5.12, 5.13 and 5.18.

5.3 Which procedure to choose? Optimizing δ.

In this section, we prove Lemmas 5.7 and 5.8. We first give the proof of Lemma 5.7. Then, with the aim of proving Lemma 5.8, we analyze functions that arise during our analysis of the running time of the algorithm.

Observe that Lemma 5.4 and Observation 5.6 together give a randomized parameterized β-approximation algorithm for $(\mathcal{G}, \Pi)$-`Del` whose running time depends on δ. More specifically, for $\delta > \frac{1}{q}$, this running is time equal to

$$\exp\left(\frac{(\delta-\beta)\cdot\ln(c) + (\beta-\alpha)\cdot\ln(1)}{\delta-\alpha}\right)^k \cdot n^{\mathcal{O}(1)} = c^{\frac{\delta-\beta}{\delta-\alpha}\cdot k}\cdot n^{\mathcal{O}(1)}.$$

Since $\frac{\delta-\beta}{\delta-\alpha} = 1 - \frac{\beta-\alpha}{\delta-\alpha}$ is increasing for $\delta \in \texttt{interval}(\alpha,\beta)$, the running time also increases for $\delta > \frac{1}{q}$. Therefore it doesn't make sense to consider values of $\delta > \frac{1}{q}$. This is why the range of δ is constrained to be less than or equal to $\frac{1}{q}$ in Lemma 5.7.

Proof. TOPROVE 16 □

Our next goal is to prove Lemma 5.8. For fixed $0 < q \leq 1$, let us define

$$\begin{aligned} h_q(\delta) := \ln\left(\tilde{f}(\delta,q)\right) &= \frac{(\delta-\beta)\cdot\ln(c) + (\beta-\alpha)\cdot\ln\left(\frac{1}{\phi(\delta,q)}\right)}{\delta-\alpha} \\ &= \frac{\delta-\beta}{\delta-\alpha}\cdot\ln(c) + \frac{\beta-\alpha}{\delta-\alpha}\cdot\ln\left(\frac{1}{\phi(\delta,q)}\right) \end{aligned}$$

We omit the subscript and write $h(\delta)$ instead of $h_q(\delta)$ whenever the values are clear from the context. Note that minimizing h is equivalent to minimizing $\tilde{f}$ as ln is a monotone increasing function, in other words

$$\min_{\delta\in\texttt{interval}(\alpha,\beta)\cap[1,\frac{1}{q}]} \tilde{f}(\delta,q) = \exp\left(\min_{\delta\in\texttt{interval}(\alpha,\beta)\cap[1,\frac{1}{q}]} h_q(\delta)\right). \tag{8}$$

Also observe that for any fixed $\delta \in \texttt{interval}(\alpha, \beta)$, $h(\delta)$ is a convex combination of $\ln(c)$ and $\ln\left(\frac{1}{\phi(\delta,q)}\right)$. To make use of this property, let us define the following function which is linear in the variable x

$$\begin{aligned} m_{\delta,q}(x) &\coloneqq \frac{\delta - x}{\delta - \alpha} \cdot \ln(c) + \frac{x - \alpha}{\delta - \alpha} \cdot \ln\left(\frac{1}{\phi(\delta, q)}\right) \\ &= \ln(c) + s_q(\delta) \cdot (x - \alpha) \end{aligned} \tag{9}$$

where

$$s_q(\delta) \coloneqq \frac{\ln\left(\frac{1}{\phi(\delta,q)}\right) - \ln(c)}{\delta - \alpha}.$$

Moreover, we have

$$h_q(\delta) = m_{\delta,q}(\beta), \tag{10}$$

and using this equivalence, the running time for an x approximation, for a fixed δ, can be stated as $d^k \cdot n^{\mathcal{O}(1)}$ where $d = \exp\Big(m_{\delta,q}(x)\Big)$.

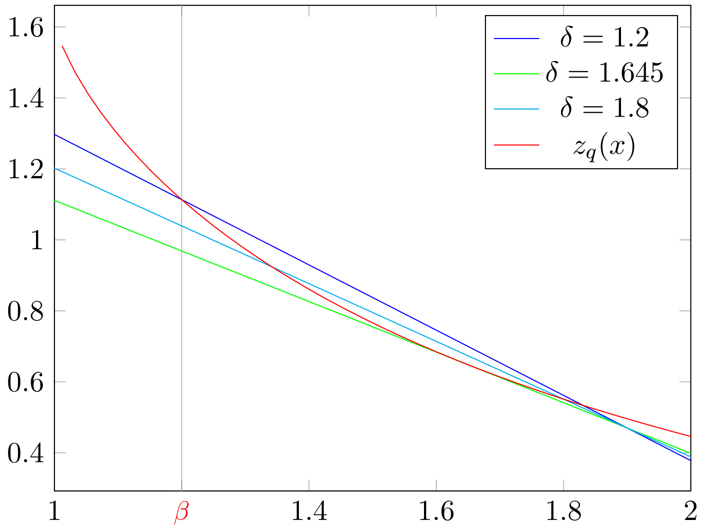

Figure 4: Comparison of the functions $m_{\delta,q}(x)$ with varying δ values (blue, green, and cyan), alongside the function $\ln\left(\frac{1}{\phi(x,q)}\right)$ (red). Recall that $m_{\delta,q}(x)$ is a linear function of x and $m_{\delta,q}(\beta) = h_q(\delta)$. Also observe that the functions $\ln\left(\frac{1}{\phi(x,q)}\right)$ and $m_{\delta,q}(x)$ meet at δ.

In the following we will demonstrate that for $0 < q < 1$, the value of δ that minimizes $h_q(\delta)$ (equivalently, $\tilde{f}(\delta, q)$) and can be found by analyzing $s_q(\delta)$.

Lemma 5.19. *Let $0 < q < 1$. It holds that*

$$\min_{\delta \in \texttt{interval}(\alpha,\beta) \cap [1, \frac{1}{q}]} h_q(\delta) = \begin{cases} \ln(c) + (\beta - \alpha) \cdot \left(\min_{\delta \in \texttt{interval}(\alpha,\beta) \cap [1,\frac{1}{q}]} s_q(\delta)\right) & \text{if } \beta > \alpha \\ \ln(c) + (\beta - \alpha) \cdot \left(\max_{\delta \in \texttt{interval}(\alpha,\beta) \cap [1,\frac{1}{q}]} s_q(\delta)\right) & \text{if } \beta < \alpha. \end{cases}$$

Proof. TOPROVE 17 ☐

Lemma 5.19 states that depending on the values of α and β, minimizing $h_q(\delta)$ is equivalent to either minimizing or maximizing $s_q(\delta)$. Therefore, in what follows, we will study the analytical properties of $s_q(\delta)$.

Lemma 5.20. *Let $0 < q \leq 1$. For $\delta \in (1, \infty) \setminus \alpha$, define*

$$\Gamma_q(\delta) \coloneqq -\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, q\right) + \alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, \frac{1}{\delta}\right) + \ln(c)$$

It holds that

$$\operatorname{sign}\left(\frac{\partial}{\partial \delta} s_q(\delta)\right) = \operatorname{sign}\Big(\Gamma_q(\delta)\Big). \tag{11}$$

Moreover, it also holds that $\frac{\partial}{\partial \delta} s_q(\delta) = 0$ if and only if $\Gamma_q(\delta) = 0$.

The proof of Lemma 5.20 can be found in Appendix D. The following lemma describes the behavior of the function $s_q(\delta)$ over specific intervals. It demonstrates that $s_q(\delta)$ is unimodal over intervals, meaning that it exhibits a strictly decreasing trend followed by a strictly increasing trend, or vice versa, depending on the interval.

Lemma 5.21. *The function $s_q(\delta)$ is strictly decreasing for $\alpha \leq \delta \leq \delta^*_{\text{right}}$ and strictly increasing for $\delta^*_{\text{right}} \leq \delta \leq \frac{1}{q}$. Moreover, if $\alpha > 1$, then the function $s_q(\delta)$ is strictly increasing for $1 \leq \delta \leq \delta^*_{\text{left}}$ and strictly decreasing for $\delta^*_{\text{left}} \leq \delta \leq \alpha$.*

The proof of Lemma 5.21 can be found in Appendix D.

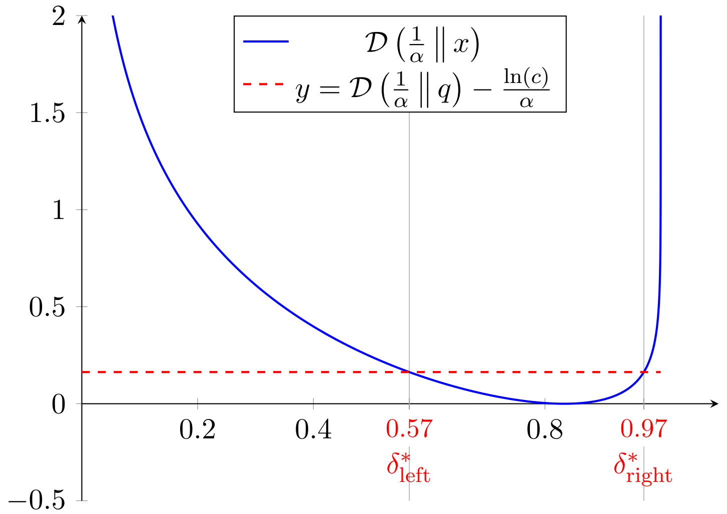

Figure 5: The plot of the function $\psi(x) = \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, x\right)$ and $y = \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, q\right) - \frac{\ln(c)}{\alpha}$ for $\alpha = 1.2, q = 0.5$ and $c = 1.1$. Note that $\psi(x)$ has a zero at $\frac{1}{\alpha}$ and it is monotone in intervals $(1, \frac{1}{\alpha}]$ and $[\frac{1}{\alpha}, 1)$.

With these definitions and results, we are now able to calculate the minimum (or maximum) of $s_q(\delta)$ over $\delta \in \texttt{interval}(\alpha, \beta) \cap [1, \frac{1}{q}]$. In Lemmas 5.22 and 5.23, we separately consider the two complementary cases based on whether α is greater than β or not.

Lemma 5.22. *Let $0 < q \leq 1$ such that $c \leq \exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, q\right)\right)$. Suppose that $\frac{1}{q} > \beta > \alpha$ and $\beta < \delta^*_{\text{right}}$. Then*

$$\left(\min_{\delta \in \texttt{interval}(\alpha,\beta) \cap [1, \frac{1}{q}]} s_q(\delta)\right) = s_q(\delta^*_{\text{right}}).$$

Proof. TOPROVE 18 □

Next, we state the analogue of Lemma 5.22 for the $\beta > \alpha$ case. The proof Lemma 5.23, which is nearly identical to the proof of Lemma 5.22, can be found in Appendix D.

Lemma 5.23. *Let* $(\alpha, \beta) \in \texttt{parameters}$, $0 < q \leq 1$, *and* $c \geq 1$ *such that* $c \leq \exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \middle\| q\right)\right)$. *Suppose that* $1 < \beta < \alpha < \frac{1}{q}$ *and* $\beta > \delta^*_{\text{left}}$. *Then*

$$\left(\max_{\delta \in \texttt{interval}(\alpha,\beta) \cap [1, \frac{1}{q}]} s_q(\delta) \right) = s_q(\delta^*_{\text{left}}).$$

Finally, we present the proof of Lemma 5.8.

Proof. TOPROVE 19 □

6 Properties of δ^*_{left} and δ^*_{right}

In this section we show properties of the functions δ^*_{left} and δ^*_{right} which has been defined in (2). We first prove Lemma 3.4 which shows the functions are well defined. Then, we provide a closed formula for δ^*_{left} and δ^*_{right} for some special cases. We also use the closed formulas to prove Theorems 3.6 and 3.7 which provide a closed formula for $\texttt{convert}\,(\alpha, \beta, c, q)$ in the these special cases.

Lemma 3.4. *For every* $c \geq 1$, $0 < q < 1$ *and* $\alpha \geq 1$ *such that* $c \leq \exp\left(\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \middle\| q\right)\right)$, *the value of* $\delta^*_{\text{right}}(\alpha, c, q)$ *is well defined. Furthermore, if* $\alpha > 1$, *then* $\delta^*_{\text{left}}(\alpha, c, q)$ *is also well defined.*

Proof. TOPROVE 20 □

Next, we first give a closed formula for δ^*_{right} in case $\alpha = 1$.

Lemma 6.1. *Let* $\alpha = 1$, $0 < q \leq 1$ *and* $c \geq 1$ *such that* $c \leq \frac{1}{q}$. *Then we have*

$$\delta^*_{\text{right}}(\alpha, c, q) = \frac{1}{q \cdot c}.$$

Proof. TOPROVE 21 □

Theorem 3.6 is a simple consequence of Lemma 6.1.

Theorem 3.6 (simple formula for $\alpha = 1$)**.** *For every* $0 < q < 1$, $1 \leq \beta \leq \frac{1}{q}$ *and* $c \geq 1$ *such that* $c \leq \exp\left(1 \cdot \mathcal{D}\left(\frac{1}{1} \middle\| q\right)\right) = \frac{1}{q}$ *it holds that*

$$\texttt{convert}\,(1, \beta, c, q) = \begin{cases} c \cdot \left(\frac{1 - c \cdot q}{1 - q}\right)^{\beta - 1} & \text{if } 1 \leq \beta < \frac{1}{q \cdot c} \\ \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \middle\| q\right)\right) & \text{if } \frac{1}{q \cdot c} \leq \beta \leq \frac{1}{q} \end{cases}$$

Proof. TOPROVE 22 □

Finally, we also provide a closed formula for δ^*_{left} in case $\alpha = 2$ and $c = 1$. Here, we rely on the fact that $\mathcal{D}\left(\frac{1}{2} \middle\| x\right)$ is symmetric around $\frac{1}{2}$.

Lemma 6.2. *Let* $\alpha = 2$, $0 < q \leq \frac{1}{\alpha} = \frac{1}{2}$ *and* $c = 1$. *Then we have*

$$\delta^*_{\text{left}}(\alpha, c, q) = \frac{1}{1 - q}.$$

Proof. TOPROVE 23 □

Theorem 3.7 (simple formula for $\alpha = 2$ and $c = 1$)**.** *Let $0 < q \leq \frac{1}{2}$ and $1 \leq \beta \leq 2$. Then it holds that*

$$\texttt{convert}\,(2, \beta, 1, q) = \begin{cases} \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \middle\| q\right)\right) & \text{if } 1 \leq \beta \leq \frac{1}{1-q} \\ \left(\frac{q}{1-q}\right)^{\beta-2} & \text{if } \frac{1}{1-q} < \beta \leq 2 \end{cases}$$

Proof. TOPROVE 24 □

7 Sampling Steps

In this section we provide sampling steps used by our applications. Sections 7.1 and 7.2 give the sampling steps for Feedback Vertex Set and Pathwidth One Vertex Deletion. In Section 7.3 we give the generic sampling step for $(\mathcal{G}, \Pi)$-`Vertex Deletion` problems in which Π is defined by a finite set of forbidden sub-hypergraphs.

7.1 Feedback Vertex Set

In this section we will prove Lemma 4.1, that is we provide a sampling step for Feedback Vertex Set (`FVS`) with success probability $\frac{1}{4}$. Let $\mathcal{G}$ denote the set of graphs and let $\Pi^{\texttt{FVS}}$ denote the set of graphs without cycles. Observe that $\Pi^{\texttt{FVS}}$ is a hereditary hypergraph property. For an input graph G, the Feedback Vertex Set problem asks whether there exists a set $S \subseteq V(G)$ of size k such that $G \setminus S \in \Pi^{\texttt{FVS}}$, i.e. $G \setminus S$ is acyclic.

The sampling step for `FVS` is given in Algorithm 5. It starts by iteratively removing vertices of degree at most 1, as described in Algorithm 4. Given an input graph G, we refer to the resulting graph from this procedure as $\texttt{Core}(G)$. It is evident that the sets of cycles in G and $\texttt{Core}(G)$ are equal because a vertex $v \in V(G)$ with degree at most 1 cannot be part of a cycle. Therefore, it holds that

$$\texttt{OPT}_{\Pi^{\texttt{FVS}}}(\texttt{Core}(G)) = \texttt{OPT}_{\Pi^{\texttt{FVS}}}(G). \tag{12}$$

After computing $\texttt{Core}(G)$), the sampling step defines a weight for every vertex. The weight of a vertex v of degree 2 (in $\texttt{Core}(G)$) is zero, and the weight of every other vertex is its degree. The algorithm returns a random vertex, so the probability of every vertex to be returned is proportional to is weight. The algorithm also handles a corner case which occurs if $\texttt{Core}(G)$ contains a cycle of vertices of degree 2 by sampling a random vertex from this cycle.

Algorithm 4 Computing the $\texttt{Core}(G)$ for a graph G

```
Input: Graph $G$
1: while $G$ has a vertex $u$ of degree at most 1 do
2:     $G \leftarrow G - \{u\}$, i.e., remove $u$ from $G$
3: return $G$
```

The proof of the next lemma is an adjustment of the arguments in [3] (see also [17]).

Lemma 7.1. *Algorithm 5 is a sampling step for* `FVS` *with success probability $\frac{1}{4}$.*

Lemma 4.1 is an immediate consequence of Lemma 7.1.

Proof. TOPROVE 25 □

Algorithm 5 Sampling Step for Feedback Vertex Set

```
Input: Graph $G$
1: $G \leftarrow \texttt{Core}(G)$
2: if $G$ has a connected component $C$ with maximum degree 2 then
3:     return a vertex $v \in C$ uniformly at random
4: For each $v \in V(G)$, define $w(v) = \begin{cases} 0 & \text{if } \deg(v) = 2 \\ \deg(v) & \text{otherwise} \end{cases}$
5: $W \leftarrow \sum_{v \in V(G)} w(v)$
6: Sample a vertex $v \in V(G)$ with respect to probabilities $\frac{w(v)}{W}$
7: return $v$
```

7.2 Pathwidth One Vertex Deletion

Let $\mathcal{G}$ denote the set of graphs, and let $\Pi^{\mathtt{POVD}}$ denote the set of graphs with pathwidth at most 1. Given a graph $G \in \mathcal{G}$ as input, the POVD (Pathwidth One Vertex Deletion) problem asks for a set $S \subseteq V(G)$ such that $G \setminus S$ belongs to $\Pi^{\mathtt{POVD}}$, i.e., $G \setminus S$ has pathwidth at most 1.

Let T_2 be the graph with 7 vertices, where we take three paths with 3 vertices each, and identify one of the degree 1 vertices of each path (see Fig. 6). Let Ω denote the set of all cycle graphs together with the graph T_2. An alternative characterization of $\Pi^{\mathtt{POVD}}$ is the following: a graph G belongs to $\Pi^{\mathtt{POVD}}$ if and only if G has no subgraph isomorphic to a graph in Ω [42, 44]. Note that the set of forbidden subgraphs in the case of Feedback Vertex Set is $\Omega \setminus \{T_2\}$, hence, it is natural to adapt the sampling step for Feedback Vertex Set to POVD.

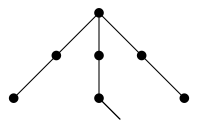

Figure 6: The graph T_2

Algorithm 6 Sampling step for Pathwidth One Vertex Deletion

```
Input: Hypergraph $G \in (\mathcal{G} \setminus \Pi^{\mathtt{POVD}})$
1: if $G$ has a subgraph $Z$ isomorphic to $T_2$ then
2:     return a vertex $v \in V(Z)$ uniformly at random
3: else if
4:     then Run Algorithm 5 (Sampling Step for Feedback Vertex Set) on $G$
```

Our sampling step for POVD, given in Algorithm 6 i, first checks whether G has a subgraph isomorphic to T_2. If this is the case, is samples a random vertex of this subgraph. If not, then G should have a subgraph isomorphic to cycle and the sampling step for `FVS` is used.

The following lemma show that Lemma 4.3 holds.

Lemma 7.2. *Algorithm 6 is a sampling step for* POVD *with success probability* $\frac{1}{7}$*.*

Proof. TOPROVE 26 □

7.3 $(\mathcal{G}, \Pi)$-Vertex Deletion for a finite set of forbidden sub-hypergraphs

We are left to prove Lemma 3.10. That is, we describe a sampling step for $(\mathcal{G}, \Pi^{\Omega})$-`Vertex Deletion` where Π^{Ω} is described by a finite set of forbidden subhypergraphs (Definition 3.8).

In the remainder of this section, let $\mathcal{G}$ be a fixed, closed set of hypergraphs, and let $\Omega = \{F_1, \ldots, F_\ell\}$ be a fixed finite set of hypergraph. Recall $\eta(\Omega) \coloneqq \max_{1 \le i \le \ell} |V(F_i)|$. The idea in the sampling step for $(\mathcal{G}, \Pi^{\Omega})$-`Vertex Deletion` is very simple, if a hypergraph $G \in \mathcal{G}$ does not belong to Π^{Ω}, then G should have a subhypergraph Z isomorphic to F_i for some $1 \le i \le \ell$. Moreover, any solution $S \in \texttt{SOL}_{\Pi^{\Omega}}(G)$ should contain a vertex from Z, otherwise $(G \setminus S) \notin \Pi^{\Omega}$. We combine these ideas in Algorithm 7.

Algorithm 7 Sampling step for $(\mathcal{G}, \Pi^{\Omega})$-`Vertex Deletion`

Configuration: A closed set of hypergraphs $\mathcal{G}$, a set of hypergraphs $\Omega = \{F_1, \ldots, F_\ell\}$, the hypergraph property Π^{Ω}

Input: $G \in \mathcal{G} \setminus \Pi^{\Omega}$

```
1: for $1 \le i \le \ell$ do
2:     for $Z \subseteq V(G)$ such that $|Z| = |F_i|$ do
3:         if $G[Z]$ is isomorphic to $F_i$ then
4:             Let $v \in Z$ be a vertex sampled uniformly at random
5:             return $v$
```

Lemma 7.3. *Algorithm 7 is a sampling step for* $(\mathcal{G}, \Pi^{\Omega})$-`Vertex Deletion` *with success probability* $\frac{1}{\eta(\Omega)}$.

We note that Lemma 3.10 follows immediately from Lemma 7.3.

Proof. TOPROVE 27 □

8 Additional Applications

In this section, following Section 4, we present additional applications of our results.

8.1 3-Hitting Set

Observe that 3-Hitting Set is equivalent to $(\mathcal{G}, \Pi)$-`Vertex Deletion` where $\mathcal{G}$ is the set of all hypergraphs with edge cardinality 3 and Π is the set of all edgeless hypergraphs. Moreover, let F be a hypergraph with a single edge of cardinality 3 and define $\Omega \coloneqq \{F\}$ such that $\eta(\Omega) = 3$. Furthermore, let Π^{Ω} be as in Definition 3.8. Note that $(\mathcal{G}, \Pi)$-`Vertex Deletion` is also equivalent to $(\mathcal{G}, \Pi^{\Omega})$-`Vertex Deletion`, because for $G \in \mathcal{G}$, it holds that $G \in \Pi$ if and only if G doesn't have an edge, i.e. there is no vertex induced subhypergraph of G isomorphic to F. By Lemma 7.3, there is a sampling step for 3-Hitting Set with success probability $\frac{1}{3}$.

We will utilize the FPT algorithm from [46] which runs in time $2.076^k \cdot n^{\mathcal{O}(1)}$.

Lemma 8.1. *There is a randomized parameterized β-approximation algorithm for* 3-Hitting Set *with running time*

$$d^k \cdot n^{\mathcal{O}(1)}$$

where we have

$$d = \begin{cases} 2.076 \cdot (0.462)^{\beta - 1} & \text{if } 1 < \beta < 1.445 \\ \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \middle\| \frac{1}{3}\right)\right) & \text{if } 1.445 \le \beta < 3. \end{cases} \tag{13}$$

Proof. TOPROVE 28 ◻

In Fig. 7, we compare our results with those from [9], [23], and [30].

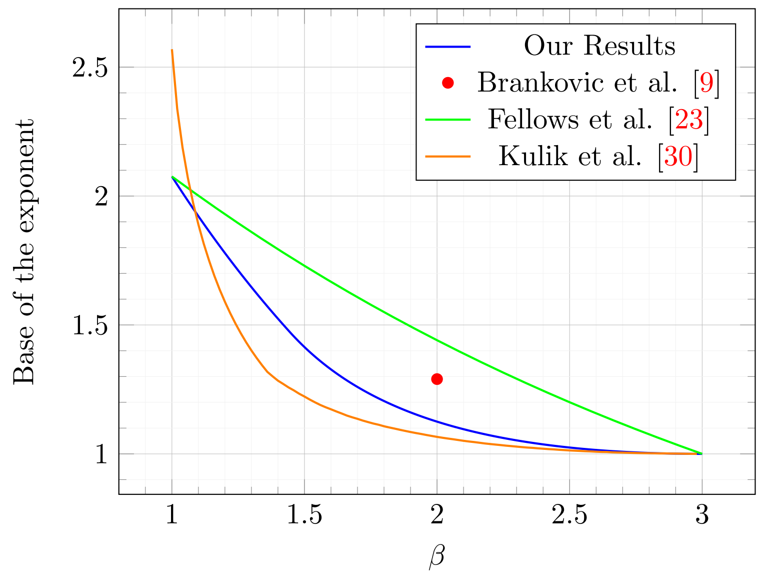

Figure 7: Comparison of the running times of various algorithms for 3-Hitting Set. The x-axis corresponds to the approximation ratio, while the y-axis corresponds to the base of the exponent in the running time. A point (β, d) in the plot describes a running time of the form $d^k \cdot n^{\mathcal{O}(1)}$ for a β-approximation. The red point corresponds to the 2-approximation algorithm from [9], with a running time of $1.29^k \cdot n^{\mathcal{O}(1)}$. Even though our result outperforms [9] and [23], it only improves upon [30] for values of β such that $\beta \lessapprox 1.16$.

8.2 4-Path Vertex Cover

Recall the definition of ℓ-Path Vertex Cover from Section 4.3. According to Lemma 3.10, there is a sampling step for
4-Path Vertex Cover with a success probability $\frac{1}{4}$. Moreover, there exists an FPT algorithm that runs in time $2.138^k \cdot n^{\mathcal{O}(1)}$ ($\alpha = 1, c = 2.138$) [14], and a 3-approximation algorithm that runs in polynomial time ($\alpha = 3, c = 1$) [12].

Lemma 8.2. *There is a randomized parameterized β-approximation algorithm for* 4-Path Vertex Cover *with running time*

$$d^k \cdot n^{\mathcal{O}(1)}$$

where we have

$$d = \begin{cases} 2.138 \cdot (0.621)^{\beta-1} & \text{if } 1 < \beta \leq 1.871 \\ \exp\left(\beta \cdot \mathcal{D}\left(\frac{1}{\beta} \,\middle\|\, \frac{1}{4}\right)\right) & \text{if } 1.871 < \beta \leq 2.357 \\ \exp\left(\frac{2.357 \cdot \mathcal{D}(0.424 \,\|\, 0.25)}{0.643} \cdot (3-\beta)\right) & \text{if } 2.357 < \beta \leq 3 \end{cases}$$

Proof. TOPROVE 29 ◻

See Fig. 8 for a plot of d in Lemma 8.2, depending on the approximation ratio β.

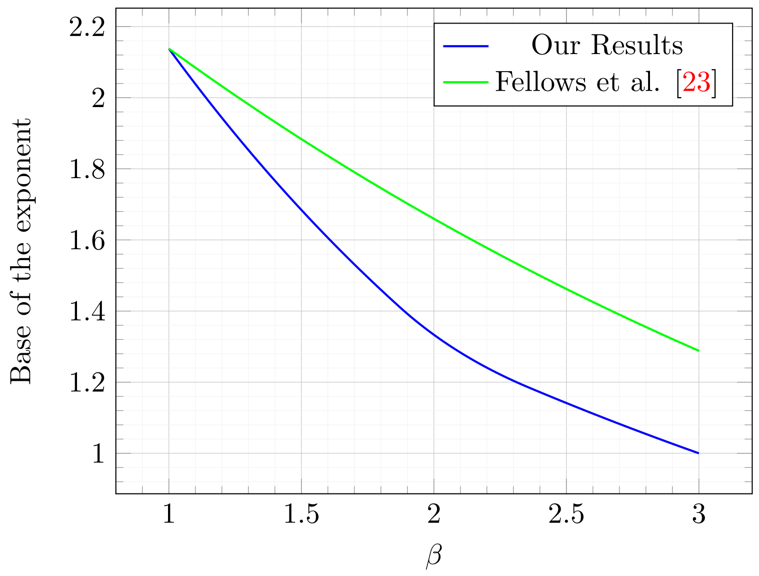

Figure 8: A plot of the running times of various algorithms for 4-Path Vertex Cover. The x-axis corresponds to the approximation ratio, while the y-axis corresponds to the base of the exponent in the running time. A point (β, d) in the plot describes a running time of the form $d^k \cdot n^{\mathcal{O}(1)}$ for a β-approximation.

8.3 Directed Feedback Vertex Set on Tournaments

In this section, we demonstrate how our techniques extend to directed graph problems. Recall that in the Directed Feedback Vertex Set on Tournaments problem, we are given a tournament graph G and we would like to find a set of vertices S such that $G \setminus S$ doesn't have any directed cycles. Although our technique normally applies to hypergraphs, we adapt it for directed graphs by defining $\mathcal{G}$ as the set of all tournament graphs. Similarly, we define the graph property Π to consist of all tournament graphs that are cycle free. Note that a tournament is acyclic if and only if it contains no directed triangle. It is not hard to show that our results for graph properties, with a finite set of forbidden graphs, also apply in this setting. We omit the technical details.

By Lemma 3.10, there is a sampling step for Directed Feedback Vertex Set on Tournaments with success probability $\frac{1}{3}$. There is also a 2-approximation algorithm that runs in polynomial time [36] ($\alpha = 2, c = 1$). Moreover, there is an FPT algorithm with running time $1.618^k \cdot n^{\mathcal{O}(1)}$ ($\alpha = 1, c = 1.618$) [31].

Lemma 8.3. *There is a randomized parameterized β-approximation algorithm for* Directed Feedback Vertex Set on Tournaments *with running time*

$$d^k \cdot n^{\mathcal{O}(1)}$$

where we have

$$d = \begin{cases} 1.618 \cdot (0.691)^{\beta-1} & \text{if } 1 < \beta \leq 1.854 \\ 0.5^{\beta-2} & \text{if } 1.854 < \beta \leq 2 \end{cases}$$

Proof. TOPROVE 30 □

See Fig. 9 for a plot of d in Lemma 8.3, depending on the approximation ratio β.

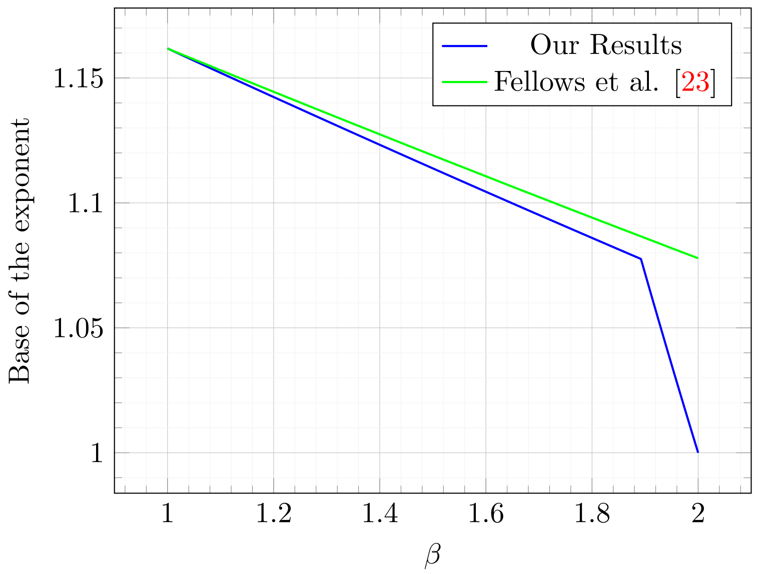

Figure 9: A plot of the running time of our algorithm for Directed Feedback Vertex Set on Tournaments. The x-axis corresponds to the approximation ratio, while the y-axis corresponds to the base of the exponent in the running time. A point (β, d) in the plot describes a running time of the form $d^k \cdot n^{\mathcal{O}(1)}$ for a β-approximation.

8.4 Vertex Cover on Graphs with Maximal Degree 3

Vertex Cover on graphs with maximal degree 3 is the restriction of the Vertex Cover problem to graphs with maximum degree 3. It can be expressed as a $(\mathcal{G}, \Pi)$-`Vertex Deletion` problem, where $\mathcal{G}$ corresponds to graphs with maximum degree 3 and the hypergraph property Π consists of all edgeless graphs. Note that Π can be described by the forbidden subgraph K_2, which is an edge that consists of two vertices. Therefore, by Lemma 3.10 there exists a sampling step with success probability $\frac{1}{2}$.

In [5], the authors present a polynomial time approximation algorithm for any approximation ratio arbitrarily close to 7/6. For simplicity, we will assume that a $\frac{7}{6}$-approximation algorithm exists ($\alpha = \frac{7}{6}, c = 1$). Note that when we consider β-approximation algorithms, we can focus on the values of β in the range $1 < \beta \leq \frac{7}{6}$ because of $\mathcal{A}_2$. Moreover, there exists an FPT algorithm with running time $1.1616^k \cdot n^{\mathcal{O}(1)}$ [48] ($\alpha = 1, c = 1.1616$).

Lemma 8.4. *There is a randomized parameterized β-approximation algorithm for* Vertex Cover on graphs with maximal degree 3 *with running time*

$$d^k \cdot n^{\mathcal{O}(1)}$$

where we have

$$d = \begin{cases} 1.1616 \cdot (0.8384)^{\beta - 1} & \text{if } 1 < \beta \leq 1.136 \\ \exp\left(\frac{1.008 \cdot \mathcal{D}(0.992 \,\|\, 0.5)}{0.158} \cdot (1.166 - \beta) \right) & \text{if } 1.136 < \beta \leq 1.166 \end{cases} \tag{14}$$

Proof. TOPROVE 31 □

See Fig. 10 for a plot of d in Lemma 8.4, depending on the approximation ratio β.

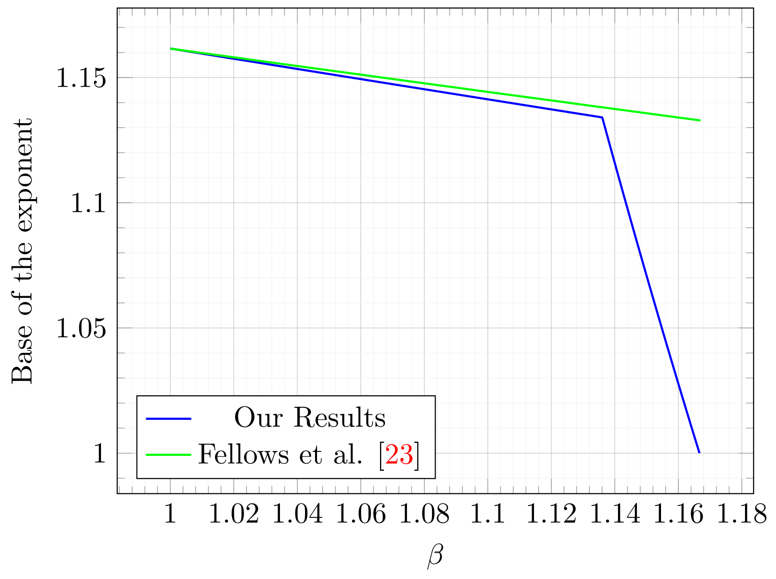

Figure 10: A plot of the running time of our algorithm for Vertex Cover on graphs with maximal degree 3. The x-axis corresponds to the approximation ratio, while the y-axis corresponds to the base of the exponent in the running time. A point (β, d) in the plot describes a running time of the form $d^k \cdot n^{\mathcal{O}(1)}$ for a β-approximation.

9 Comparison to Fidelity Preserving Transformations

In this section we will prove Lemma 3.12 which implies that Sampling with a Black Box provides better running time than Fidelity Preserving Transformations. Here we state the lemma once again for completeness.

Lemma 3.12. *For every $\eta \in \mathbb{N}$ such that $\eta \geq 2$, $1 < c < \eta$ and $1 < \beta < \eta$ it holds that*

$$\texttt{convert}\left(1, \beta, c, \frac{1}{\eta}\right) < c^{\frac{\eta-\beta}{\eta-1}}.$$

Proof. TOPROVE 32 □

10 Discussion

In this paper we presented Sampling with a Black Box, a simple and generic technique for the design of parameterized approximation algorithms for vertex deletion problems. The technique relies on sampling steps, polynomial time algorithms which return a random vertex whose removal reduces the optimum by one, with some success probability q. The technique combines the sampling step with existing parameterized and approximation algorithms to derive efficient parameterized approximation algorithms. We provide application for various problems, such as Feedback Vertex Set, ℓ-Path Vertex Cover, d-Hitting Set and Directed Feedback Vertex Set on Tournaments.

We point out two directions for follow up works:

- While Sampling with a Black Box provides faster parameterized approximation algorithms for multiple problems, it does not provide a significant improvement for problems which has been extensively studied from this angle, such as Vertex Cover and 3-Hitting Set [9, 10, 23, 30]. This can be potentially improved by replacing sampling steps with a more generic procedure which can apply randomized branching rules [30]. Initial research suggests this could result in better running times for several problems.

.

- Our results are focused on *unweighted* vertex deletion problems. Vertex deletion problems can be naturally generalized for the weighted setting, in which each vertex $v \in V(G)$ has a weight $w(v)$ and the objective is to find a set $S \subseteq V(G)$ of minimum *weight* $\sum_{v \in S} w(v)$ such that $G \setminus S$ satisfies the property Π. In particular, parameterized approximation algorithms for the special case of Weighted Vertex Cover has been recently considered in [38]. It would be interesting to adjust Sampling with a Black Box to the weighted setting. For example, such a result may improve the running times of [38] for Weighted Vertex Cover.

  In [22] the authors applied a rounding procedure over weights in order to utilize the (approximate) monotone local search technique of [13] in a weighted setting. Intuitively, a similar approach may also be useful for Sampling with a Black Box.

In this paper we designed *exponential time* parameterized approximation algorithms for vertex deletion problems. For many of the considered problems, such as Vertex Cover and 3-Hitting Set, it is known that, assuming the Exponential Time Hypothesis (ETH), there is no sub-exponential time parameterized (exact) algorithms (see, e.g. [17]). However, it is less clear whether the considered problems admit sub-exponential time parameterized approximation algorithms for approximation ratios close to 1. The existence of strictly sub-exponential parameterized approximation algorithms for Vertex Cover for certain approximation ratio has been rules out, assuming ETH, in [7]. However, we are not aware of a result which rules out a $c^{o(k)} \cdot n^{\mathcal{O}(1)}$ parameterized $(1+\varepsilon)$-approximation for Vertex Cover (or other vertex deletion problem) for a some constant $\varepsilon > 0$. It would be interesting to explore whether recent tools in parameterized inapproximability, possibly together with the stronger Gap Exponential Time Hypothesis (GAP-ETH) [18, 41], can lead to such a result.

References

[1] Fateme Abbasi, Sandip Banerjee, Jarosław Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation schemes for clustering with general norm objectives. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1377–1399. IEEE, 2023.

[2] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-Approximation Algorithm for the Undirected Feedback Vertex Set Problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, January 1999.

[3] Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *Journal of Artificial Intelligence Research*, 12(1):219–234, May 2000.

[4] Rémy Belmonte, Michael Lampis, and Valia Mitsou. Parameterized (approximate) defective coloring. *SIAM Journal on Discrete Mathematics*, 34(2):1084–1106, 2020.

[5] P. Berman and T. Fujito. On Approximation Properties of the Independent Set Problem for Low Degree Graphs. *Theory of Computing Systems*, 32(2):115–132, April 1999.

[6] Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3), mar 2021.

[7] Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th Paschos. On subexponential and fpt-time inapproximability. *Algorithmica*, 71:541–565, 2015.

[8] Nicolas Bourgeois, Bruno Escoffier, and Vangelis Th Paschos. Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms. *Discrete Applied Mathematics*, 159(17):1954–1970, 2011.

[9] Ljiljana Brankovic and Henning Fernau. Parameterized Approximation Algorithms for Hitting Set. In Roberto Solis-Oba and Giuseppe Persiano, editors, *Approximation and Online Algorithms*, pages 63–76, Berlin, Heidelberg, 2012. Springer.

[10] Ljiljana Brankovic and Henning Fernau. A novel parameterised approximation algorithm for minimum vertex cover. *Theoretical Computer Science*, 511:85–108, 2013.

[11] Boštjan Brešar, František Kardoš, Ján Katrenič, and Gabriel Semanišin. Minimum k-path vertex cover. *Discrete Applied Mathematics*, 159(12):1189–1195, July 2011.

[12] Elias Camby, Jean Cardinal, Michaël Chapelle, Samuel Fiorini, and Gwenaël Joret. A primal-dual 3-approximation algorithm for hitting 4-vertex paths. In *Proceedings of the 9th International Colloquium on Graph Theory and Combinatorics (ICGT 2014)*, page 61, Grenoble, France, 2014.

[13] Barış Can Esmer, Ariel Kulik, Dániel Marx, Daniel Neuen, and Roohani Sharma. Optimally Repurposing Existing Algorithms to Obtain Exponential-Time Approximations. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 314–345. Society for Industrial and Applied Mathematics, January 2024.

[14] Radovan Červený and Ondřej Suchý. Generating Faster Algorithms for d-Path Vertex Cover. In Daniël Paulusma and Bernard Ries, editors, *Graph-Theoretic Concepts in Computer Science*, pages 157–171, Cham, 2023. Springer Nature Switzerland.

[15] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-exponential time hypothesis to fixed parameter tractable inapproximability: Clique, dominating set, and more. *SIAM Journal on Computing*, 49(4):772–810, 2020.

[16] Yijia Chen, Yi Feng, Bundit Laekhanukit, and Yanlin Liu. Simple combinatorial construction of the $k^{o(1)}$ -lower bound for approximating the parameterized k-clique. *arXiv preprint arXiv:2304.07516*, 2023.

[17] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, Cham, 2015.

[18] Irit Dinur. Mildly exponential reduction from gap-3sat to polynomial-gap label-cover. In *Electronic colloquium on computational complexity ECCC; research reports, surveys and books in computational complexity*, page 128, 2016.

[19] Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. Budgeted Matroid Maximization: a Parameterized Viewpoint. In *Proc. IPEC 2023*, pages 13:1–13:17, 2023.

[20] Pavel Dvorák, Andreas Emil Feldmann, Dusan Knop, Tomás Masarík, Tomas Toufar, and Pavel Veselỳ. Parameterized approximation schemes for steiner trees with small number of steiner vertices. In *Proc. STACS*, 2018.

[21] Barış Can Esmer, Ariel Kulik, Dániel Marx, Daniel Neuen, and Roohani Sharma. Faster Exponential-Time Approximation Algorithms Using Approximate Monotone Local Search. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *Proc. ESA*, 2022.

[22] Barış Can Esmer, Ariel Kulik, Dániel Marx, Daniel Neuen, and Roohani Sharma. Approximate monotone local search for weighted problems. In Neeldhara Misra and Magnus Wahlström, editors, *18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*, volume 285 of *Leibniz International Proceedings in Informatics (Lipics)*, pages 17:1–17:23, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[23] Michael R Fellows, Ariel Kulik, Frances Rosamond, and Hadas Shachnai. Parameterized approximation via fidelity preserving transformations. *Journal of Computer and System Sciences*, 93:30–40, 2018.

[24] Fedor V Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *Journal of the ACM (JACM)*, 66(2):1–23, 2019.

[25] Fabrizio Grandoni, Stefan Kratsch, and Andreas Wiese. Parameterized Approximation Schemes for Independent Set of Rectangles and Geometric Knapsack. In *Proc. ESA*, 2019.

[26] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Parameterized inapproximability hypothesis under ETH. In *Proc. STOC (to appear)*, 2024.

[27] Tanmay Inamdar, Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. Parameterized Complexity of Fair Bisection: (FPT-Approximation meets Unbreakability). In *Proc. ESA*, 2023.

[28] Satyabrata Jana, Daniel Lokshtanov, Soumen Mandal, Ashutosh Rai, and Saket Saurabh. Parameterized Approximation Scheme for Feedback Vertex Set. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 56:1–56:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[29] CS Karthik and Subhash Khot. Almost polynomial factor inapproximability for parameterized k-clique. In *37th Computational Complexity Conference (CCC 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2022.

[30] Ariel Kulik and Hadas Shachnai. Analysis of two-variable recurrence relations with application to parameterized approximations. In *Proc. FOCS*, pages 762–773, 2020.

[31] Mithilesh Kumar and Daniel Lokshtanov. Faster Exact and Parameterized Algorithm for Feedback Vertex Set in Tournaments. *LIPIcs, Volume 47, STACS 2016*, 47:49:1–49:13, 2016.

[32] Michael Lampis. Parameterized approximation schemes using graph widths. In *International Colloquium on Automata, Languages, and Programming*, pages 775–786. Springer, 2014.

[33] Jason Li and Jesper Nederlof. Detecting Feedback Vertex Sets of Size k in O⋆ (2.7k) Time. *ACM Transactions on Algorithms*, 18(4):34:1–34:26, October 2022.

[34] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Constant approximating parameterized k-setcover is W[2]-hard. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3305–3316. SIAM, 2023.

[35] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Improved hardness of approximating k-clique under eth. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, 2023.

[36] Daniel Lokshtanov, Pranabendu Misra, Joydeep Mukherjee, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. 2-Approximating Feedback Vertex Set in Tournaments. *ACM Transactions on Algorithms*, 17(2):1–14, April 2021.

[37] Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. *SIAM Journal on Computing*, 2022.

[38] Soumen Mandal, Pranabendu Misra, Ashutosh Rai, and Saket Saurabh. Parameterized approximation algorithms for weighted vertex cover. In *Latin American Symposium on Theoretical Informatics*, pages 177–192. Springer, 2024.

[39] Pasin Manurangsi. A Note on Max k-Vertex Cover: Faster FPT-AS, Smaller Approximate Kernel and Improved Approximation. In *Proc. SOSA*, 2019.

[40] Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *Proc. SODA*, pages 62–81. SIAM, 2020.

[41] Pasin Manurangsi and Prasad Raghavendra. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 78:1–78:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[42] Geevarghese Philip, Venkatesh Raman, and Yngve Villanger. A Quartic Kernel for Pathwidth-One Vertex Deletion. In Dimitrios M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science*, pages 196–207, Berlin, Heidelberg, 2010. Springer.

[43] MTCAJ Thomas and A Thomas Joy. Elements of information theory, 2006.

[44] Dekel Tsur. Faster algorithm for pathwidth one vertex deletion. *Theoretical Computer Science*, 921:63–74, June 2022.

[45] Jianhua Tu and Wenli Zhou. A factor 2 approximation algorithm for the vertex cover P_3 problem. *Information Processing Letters*, 111(14):683–686, July 2011.

[46] Magnus Wahlström. *Algorithms, Measures and Upper Bounds for Satisfiability and Related Problems.* Department of Computer and Information Science, Linköpings universitet, Linköping, 2007.

[47] Michał Włodarczyk. Parameterized inapproximability for steiner orientation by gap amplification. In *Proc. ICALP*, 2020.

[48] Mingyu Xiao. A Note on Vertex Cover in Graphs with Maximum Degree 3. In My T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics*, pages 150–159, Berlin, Heidelberg, 2010. Springer.

A Problem Definitions

In this section we will give formal definitions of the problems mentioned in this paper. Recall that a feedback vertex set of a graph G is a subset of its vertices $S \subseteq V(G)$ such that $G \setminus S$ is acyclic.

Feedback Vertex Set (`FVS`)
Input: A graph G, an integer k.
Question: Does G have a feedback vertex set of size at most k?
$(\mathcal{G}, \Pi)$-`Del` **Equivalence:** $\mathcal{G}$ is the set of all graphs, Π is the set of all graphs that have no cycles.

Pathwidth One Vertex Deletion (`POVD`)
Input: A graph G, an integer k.
Question: Is there a set of vertices $S \subseteq V(G)$ of size at most k such that $G \setminus S$ has pathwidth at most 1?
$(\mathcal{G}, \Pi)$-`Del` **Equivalence:** $\mathcal{G}$ is the set of all graphs and Π is the set of all graphs with pathwidth at most 1 (i.e. the set of caterpillar graphs).

ℓ-Path Vertex Cover
Input: A graph G, an integer k.
Question: Is there a set of vertices $S \subseteq V(G)$ of size at most k such that every path of length ℓ contains a vertex from S?
$(\mathcal{G}, \Pi)$-`Del` **Equivalence:** $\mathcal{G}$ is the set of all graph and Π is the set of graphs with maximum path length $\ell - 1$.

d-Hitting Set
Input: A universe U, a set system $\mathcal{S}$ over U where each set $S \in \mathcal{S}$ has size at most d, an integer k.
Question: Is there a set $W \subseteq U$ of size at most k such that $W \cap S \neq \emptyset$ for all $S \in \mathcal{S}$?
$(\mathcal{G}, \Pi)$-`Del` **Equivalence:** $\mathcal{G}$ is the set of all hypergraphs with edge cardinality 3 and Π is the set of all edgeless hypergraphs.

Directed Feedback Vertex Set on Tournaments
Input: A tournament graph G, an integer k.
Question: Is there a set of vertices $S \subseteq G$ of size at most k such that $G \setminus S$ has no directed cycles?
$(\mathcal{G}, \Pi)$-`Del` **Equivalence:** $\mathcal{G}$ is the set of all tournament graphs and Π is the set of all tournaments that are cycle free.

Vertex Cover on graphs with maximal degree 3
Input: A graph G with maximum degree 3, an integer k
Question: Is there a set of vertices $S \subseteq V(G)$ of size at most k such that such that every edge of G has at least one vertex from S?
$(\mathcal{G}, \Pi)$-`Del` **Equivalence:** $\mathcal{G}$ is the set of all graphs with maximum degree 3 and Π is the set of all edgeless graphs.

B Probabilistic Concepts

Let $\mathcal{D}(x \,\|\, y)$ denote the Kullback-Leibler divergence between two Bernoulli distributions with parameters x and y, i.e.

$$\begin{aligned}\mathcal{D}(x \,\|\, y) &:= x \cdot \ln\left(\frac{x}{y}\right) + (1-x) \cdot \ln\left(\frac{1-x}{1-y}\right) \\ &= x \cdot \ln\left(\frac{x}{1-x} \cdot \frac{1-y}{y}\right) + \ln\left(\frac{1-x}{1-y}\right). \end{aligned} \tag{15}$$

In our analysis of procedures, we need the following technical result which is a special case of Theorem 11.1.4 in [43].

Theorem B.1. *For any $0 \le p \le 1$ and integer $x \ge 1$, let $\texttt{binom}(x,p)$ denote the binomial random variable with success probability p and number of trials x. For any $0 \le y \le x$, it holds that*

$$\Pr\left(\texttt{binom}(x,p) \ge y\right) \ge (x+1)^{-2} \cdot \exp\left(-x \cdot \mathcal{D}\left(\frac{y}{x} \,\middle\|\, p\right)\right).$$

Moreover, we also need the following lemma, which we use to lower bound the tail probability of the sum of certain random variables.

Lemma B.2. *Let $X_1, \ldots, X_n \in \{0,1\}$ be random variables, let $p \in [0,1]$ and assume that*

$$\Pr\left(X_j = 1 \,|\, X_1 = x_1, \ldots, X_{j-1} = x_{j-1}\right) \ge p$$

for all $j \in [n]$ and $(x_1, \ldots, x_{j-1}) \in \{0,1\}^{j-1}$. Then, for every $w \in \mathbb{R}$ it holds that

$$\Pr\left(\sum_{j=1}^{n} X_j \ \ge\ w\right) \ge \Pr(\texttt{binom}(n,p) \ \ge\ w).$$

Proof. TOPROVE 33 ☐

We can combine Theorem B.1 and Lemma B.2 to obtain the following result.

Lemma 5.15. *Let $\delta, t \ge 1$ be integers, $\nu \in (0,1]$ be a real number and $\xi_1, \ldots, \xi_{\lfloor \delta \cdot t \rfloor} \in \{0,1\}$ be random variables such that*

$$\Pr\left(\xi_j = 1 \,|\, \xi_1 = x_1, \ldots, \xi_{j-1} = x_{j-1}\right) \ge \nu$$

for all $1 \le j \le \lfloor \delta \cdot t \rfloor$ and $(x_1, \ldots, x_{j-1}) \in \{0,1\}^{j-1}$. Then, there exist integers r and T that depend on δ, such that for $t \ge T$ it holds that

$$\Pr\left(\sum_{j=1}^{\lfloor \delta \cdot t \rfloor} \xi_j \ge t\right) \ge (\delta \cdot t + 1)^{-r} \cdot \exp\left(-\delta \cdot \mathcal{D}\left(\frac{1}{\delta} \,\middle\|\, \nu\right)\right)^t.$$

Proof. TOPROVE 34 ☐

C Technical Claims

Lemma C.1. *It holds that*

$$\frac{\partial}{\partial\,\delta}\left(\ln\left(\frac{1}{\phi(\delta,q)}\right)\right)=\ln\left(\frac{1-\frac{1}{\delta}}{1-q}\right) \tag{16}$$

and

$$\frac{\partial}{\partial\,\delta}\,s_q(\delta)=\frac{(\alpha-1)\cdot\ln\left(\frac{1-q}{1-\frac{1}{\delta}}\right)+\ln\left(\delta\cdot q\right)+\ln(c)}{(\delta-\alpha)^2}. \tag{17}$$

Proof. TOPROVE 35 □

Next, we state an equivalence which will be used frequently in the following section.

Lemma C.2. *It holds that*

$$\exp\Big(\ln(c)+(\beta-\alpha)\cdot s_q(\beta)\Big)=\exp\left(\beta\cdot\mathcal{D}\left(\frac{1}{\beta}\,\middle\|\,q\right)\right).$$

Proof. TOPROVE 36 □

D Omitted Proofs

Lemma 5.14. *Let* Π *be a hereditary hypergraph property and* G *be a hypergraph. For any* $v\in V(G)$*, it holds that*

$$0\le \mathtt{OPT}_\Pi(G)-\mathtt{OPT}_\Pi(G\setminus v)\le 1.$$

Proof. TOPROVE 37 □

Now, we provide the previously omitted proof of Lemma 5.21. For the sake of completeness, we restate the lemma below.

Lemma 5.21. *The function* $s_q(\delta)$ *is strictly decreasing for* $\alpha\le\delta\le\delta^*_{\text{right}}$ *and strictly increasing for* $\delta^*_{\text{right}}\le\delta\le\frac{1}{q}$*. Moreover, if* $\alpha>1$*, then the function* $s_q(\delta)$ *is strictly increasing for* $1\le\delta\le\delta^*_{\text{left}}$ *and strictly decreasing for* $\delta^*_{\text{left}}\le\delta\le\alpha$*.*

Proof. TOPROVE 38 □

Next we present the missing proof of Lemma 5.23. For completeness, we again state the lemma here.

Lemma 5.23. *Let* $(\alpha,\beta)\in\mathtt{parameters}$*,* $0<q\le 1$*, and* $c\ge 1$ *such that* $c\le\exp\left(\alpha\cdot\mathcal{D}\left(\frac{1}{\alpha}\,\middle\|\,q\right)\right)$*. Suppose that* $1<\beta<\alpha<\frac{1}{q}$ *and* $\beta>\delta^*_{\text{left}}$*. Then*

$$\left(\max_{\delta\in\mathtt{interval}(\alpha,\beta)\cap[1,\frac{1}{q}]} s_q(\delta)\right)=s_q(\delta^*_{\text{left}}).$$

Proof. TOPROVE 39 □

Let us now give the omitted proof of Lemma 5.20.

Lemma 5.20. *Let $0 < q \leq 1$. For $\delta \in (1, \infty) \setminus \alpha$, define*

$$\Gamma_q(\delta) \coloneqq -\alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, q\right) + \alpha \cdot \mathcal{D}\left(\frac{1}{\alpha} \,\middle\|\, \frac{1}{\delta}\right) + \ln(c)$$

It holds that

$$\operatorname{sign}\left(\frac{\partial}{\partial\,\delta}\, s_q(\delta)\right) = \operatorname{sign}\Big(\Gamma_q(\delta)\Big). \tag{11}$$

Moreover, it also holds that $\frac{\partial}{\partial\,\delta}\, s_q(\delta) = 0$ if and only if $\Gamma_q(\delta) = 0$.

Proof. TOPROVE 40 □