A Note on Max k-Vertex Cover: Faster FPT-AS, Smaller Approximate Kernel and Improved Approximation

Pasin Manurangsi[1]
University of California, Berkeley, CA, USA
pasin@berkeley.edu

Abstract

In Maximum k-Vertex Cover (Max k-VC), the input is an edge-weighted graph G and an integer k, and the goal is to find a subset S of k vertices that maximizes the total weight of edges covered by S. Here we say that an edge is covered by S iff at least one of its endpoints lies in S.

We present an FPT approximation scheme (FPT-AS) that runs in $(1/\varepsilon)^{O(k)}\mathsf{poly}(n)$ time for the problem, which improves upon Gupta, Lee and Li's $(k/\varepsilon)^{O(k)}\mathsf{poly}(n)$-time FPT-AS [30, 29]. Our algorithm is simple: just use brute force to find the best k-vertex subset among the $O(k/\varepsilon)$ vertices with maximum weighted degrees.

Our algorithm naturally yields an (efficient) approximate kernelization scheme of $O(k/\varepsilon)$ vertices; previously, an $O(k^5/\varepsilon^2)$-vertex approximate kernel is only known for the unweighted version of Max k-VC [43]. Interestingly, this also has an application outside of parameterized complexity: using our approximate kernelization as a preprocessing step, we can directly apply Raghavendra and Tan's SDP-based algorithm for 2SAT with cardinality constraint [52] to give an 0.92-approximation algorithm for Max k-VC in polynomial time. This improves upon the best known polynomial time approximation algorithm of Feige and Langberg [23] which yields $(0.75 + \delta)$-approximation for some (small and unspecified) constant $\delta > 0$.

We also consider the minimization version of the problem (called Min k-VC), where the goal is to find a set S of k vertices that minimizes the total weight of edges covered by S. We provide a FPT-AS for Min k-VC with similar running time of $(1/\varepsilon)^{O(k)}\mathsf{poly}(n)$. Once again, this improves on a $(k/\varepsilon)^{O(k)}\mathsf{poly}(n)$-time FPT-AS of Gupta et al. On the other hand, we show, assuming a variant of the Small Set Expansion Hypothesis [50] and $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, that there is no polynomial size approximate kernelization for Min k-VC for any factor less than two.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms, Mathematics of computing → Approximation algorithms

Keywords and phrases Maximum k-Vertex Cover, Minimum k-Vertex Cover, Approximation Algorithms, Fixed Parameter Algorithms, Approximate Kernelization

Digital Object Identifier 10.4230/OASIcs.SOSA.2019.15

Related Version A full version of the paper is available at [44], `https://arxiv.org/abs/1810.03792`.

1 Introduction

In the *Vertex Cover* problem, we are given a graph G and an integer k, and the goal is to determine whether there is a set S of k vertices that covers all the edges, where the edge is said to be covered by S if at least one of its endpoints lies in S. Vertex Cover is a classic

[1] Supported by NSF under Grants No. CCF 1655215 and CCF 1815434.

graph problem and is among Karp's original list of 21 NP-complete problems [37]. This NP-hardness has led to studies of variants of the problems. One such direction is to consider the optimization versions of the problem. Arguably, the two most natural optimization formulations of Vertex Cover are the *Minimum Vertex Cover (Min VC)* problem, where the constraint that every edge is covered is treated as a hard constraint and the goal is to find S with smallest size that satisfies this, and the *Maximum k-Vertex Cover (Max k-VC)* problem[2], where the cardinality constraint $|S| = k$ is treated as a hard constraint and the goal is to find such S that covers as many edges as possible.

Both problems have been thoroughly studied in the approximation algorithms and hardness of approximation literature. Min VC admits a simple greedy 2-approximation algorithm[3], which has been known since the seventies (see e.g. [26]). The approximation ratio has subsequently been slightly improved [9, 47] and, currently, the best known approximation ratio in polynomial time is $(2 - 1/O(\sqrt{\log n}))$ [36]. There has also been a number of works on hardness of approximation of Vertex Cover [11, 34, 22, 41, 8, 39, 40]. The best known NP-hardness of approximation for Min VC, established in the recent works that resolve the (imperfect) 2-to-1 conjecture [39, 20, 21, 40], has a factor of $(\sqrt{2}-\varepsilon)$ for any $\varepsilon > 0$. Assuming the Unique Games Conjecture (UGC) [38], the inapproximability ratio can be improved to $(2 - \varepsilon)$ for any $\varepsilon > 0$ [41, 8], which is tight up to lower order terms.

Unlike Min VC, tight approximability results for Max k-VC are not known (even assuming UGC). In particular, on the algorithmic front, the best known efficient approximation algorithm due to Feige and Langberg [23] yields a $(0.75 + \delta)$-approximation for the problem, where $\delta > 0$ is a (small) constant. This was an improvement over an earlier 0.75-approximation algorithm of Ageev and Sviridenko [2], which in turn improved upon the simple greedy algorithm that yields $(1 - 1/e)$-approximation for the problem [35]. (See also [33, 32, 31] where improvements have been made for certain ranges of k and n.) On the hardness of approximation front, it is known that the problem is NP-hard to approximate to within $(1 + \delta)$ factor for some (small) $\delta > 0$ [49]. Moreover, it follows from a result of Austrin, Khot and Safra [7] that it is UG-hard to approximate the problem to within a factor of 0.944. (See Appendix A of the full version [44].) This leaves quite a large gap between the upper and lower bounds, even assuming the UGC.

Approximability is not the only aspect of Vertex Cover and its variants that has been thoroughly explored: its parameterized complexity is also a well-studied subject. Recall that an algorithm is said to be *fixed-parameter (FPT)* with respect to parameter k if it runs in time $f(k) \cdot \mathsf{poly}(n)$ for some function f, where n is the size of the input. An FPT algorithm (with running time $k^{O(k)} \cdot \mathsf{poly}(n)$) was first devised for Vertex Cover by Buss and Goldsmith [14]. Since then, many different FPT algorithms have been discovered for Vertex Cover; to the best of our knowledge, the fastest known algorithm is that of Chen, Kanj and Xia [17], which runs in $1.2738^k \cdot \mathsf{poly}(n)$ time.

Notice that an FPT algorithm for Vertex Cover can also be adapted to solve Min VC in FPT time parameterized by the optimal solution size, by running the Vertex Cover algorithm for $k = 1, 2, \ldots$ until it finds the size of the optimal solution. On the other hand, Max k-VC

[2] Max k-VC and Min k-VC (which will be introduced below) are sometimes referred to as the Max Partial Vertex Cover and Min Partial Vertex Cover respectively. However, we decide against calling them as such to avoid ambiguity since Partial Vertex Cover has also used to refer to a different variant of Vertex Cover (see e.g. [10]).

[3] Throughout this note, we use the convention that the approximation ratio is the worst case ratio between the cost of the output solution and the optimum. In other words, the approximation ratios for maximization problems will be at most one, whereas the approximation ratios for minimization problems will be at least one.

is unlikely to admit an FPT algorithm, as it is W[1]-hard [28]. Circumventing this hardness, Marx [46] designed an *FPT approximation scheme (FPT-AS)*, which is an FPT algorithm that can achieve approximation ratio $(1-\varepsilon)$ (or $(1+\varepsilon)$ for minimization problems) for any $\varepsilon > 0$, for Max k-VC. In particular, his algorithm runs in time $(k/\varepsilon)^{O(k^3/\varepsilon)} \cdot \mathsf{poly}(n)$. This should be contrasted with the aforementioned fact that Max k-VC does not admit a PTAS unless P = NP. Recently, the FPT-AS has been sped up by Gupta, Lee and Li [30, 29][4] to run in time $(k/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$.

FPT algorithms are intimately connected to the notion of kernel. A *kernelization algorithm* (or *kernel*) of a parameterized problem is a polynomial time algorithm that, given an instance (I, k), produces another instance (I', k') such that the size of the new instance $|I'|$ and the new parameter k' are both bounded by $g(k)$ for some function g. It is well known that a parameterized problem admits a kernel if and only if it admits FPT algorithms [15]. Once again, many kernels are known for Vertex Cover (see e.g. [1] and references therein). On the other hand, the W[1]-hardness of Max k-VC means that it does not admit a kernel unless W[1] = FPT.

Recently, there have been attempts to make the concept of kernelization compatible with approximation algorithms [24, 43]. In this note, we follow the notations defined by Lokshtanov et al. [43]. For our purpose, it suffices to define an *α-approximate kernel* for an parameterized optimization problem as a pair of polynomial time algorithms $\mathcal{A}$, the *reduction algorithm*, and $\mathcal{B}$, the *solution lifting algorithm*, such that (i) given an instance (I, k), $\mathcal{A}$ produces another instance (I', k') such that $|I'|, k'$ are bounded by $g(k)$ for some g and (ii) given an β-approximate solution s' for (I', k'), $\mathcal{B}$ produces a solution s of (I, k) such that s is an $(\alpha\beta)$-approximate solution[5] for (I, k). Akin to (exact) kernelization, Lokshtanov et al. [43] shows that a decidable parameterized optimization problem admits α-approximate kernel if and only if it admits an FPT α-approximation algorithm. (We refer interested readers to Section 2.1 of [43] for more details.) In light of Marx's algorithm for Max k-VC [46], this immediately implies that Max k-VC admits $(1-\varepsilon)$-approximate kernel for any $\varepsilon > 0$. Lokshtanov et al. [43] made this bound more specific, by showing that the insights from Marx's work can be turned into an $(1-\varepsilon)$-approximate kernel where the number of vertices in the new instance is at most $O(k^5/\varepsilon^2)$.

Minimum k-Vertex Cover. We will also consider the minimization variant of the Min k-VC, which we call *Minimum k-Vertex Cover (Min k-VC)*. The goal of this problem is to find a subset of k vertices that *minimizes* the number of edges covered. Note that this is *not* a natural relaxation of Vertex Cover and is in fact more closely related to edge expansion problems. (See [25] and discussion therein for more information.) The greedy algorithm that picks k vertices with minimum degrees yields a 2-approximation. Gandhi and Kortsarz [25] showed that this is likely tight: assuming the Small Set Expansion Conjecture [50], it is hard to approximate Min k-VC to within $(2-\varepsilon)$ factor for any $\varepsilon > 0$. As for its parameterized complexity, similar to Max k-VC, Min k-VC is W[1]-hard [28] and admits an FPT-AS with running time $(k/\varepsilon)^{O(k)}$ [30, 29].

[4] In fact, Gupta et al.gives an FPT-AS for Min k-VC; it is trivial to see that their algorithm works for Max k-VC as well.

[5] We use a similar convention here as our convention for approximation ratios. That is, $\alpha \leqslant 1$ for maximization problems and $\alpha \geqslant 1$ for minimization problems. Note that this is not the same as in [43] where $\alpha \geqslant 1$ in both cases; nevertheless, it is simple to see that these different conventions do not effect any of the results.

Weight vs Unweighted. All results stated above are for unweighted graphs. The natural extensions of Max k-VC (resp. Min k-VC) to edge-weighted graphs ask to find subsets of vertices of size k that maximizes (resp. minimizes) the total weight of the edges covered. To avoid confusion, we refer to these weighted variants explicitly as Weighed Max k-VC and Weight Min k-VC. Clearly, since these problems are more general than the unweighted ones, the lower bounds above (including inapproximability results and W[1]-hardness) applies immediately. It is also quite simple to check that all aforementioned polynomial time approximation algorithms for the unweighted case extends naturally to the weighted setting too. The FPT-ASes are slightly trickier, but Gupta et al. [30] provide an argument discretizing the weights and extend their FPT-ASes to the weighted case with similar time complexity. It is also possible to apply this argument to Lokshtanov et al.'s [43] approximate kernel, although it would result in a graph of $O(k^7/\varepsilon^4)$ vertices instead of $O(k^5/\varepsilon^2)$ for the unweighted case.

1.1 Our Results

For convenience, all our results stated below are for the weighted version of the problems, and moreover we allow self-loops in the input graph. This is the most general version of the problem and, hence, the algorithmic results below apply directly to the unweighted case (nd the weighted simple graph case. We also note that this choice is partly motivated by the fact that in some applications, such Gupta et al.'s [30, 29] algorithms for Minimum k-Cut, this full generality is needed. (Unfortunately, our result does not imply faster algorithms for Minimum k-Cut, as the bottlenecks of Gupta et al.'s approach is elsewhere[6].)

We remark that, while the algorithmic results apply directly to the more restricted version, the approximate kernel does *not*. This is because, in a more restricted version (e.g. unweighted) of the problems, the instance output by the reduction algorithm is also more restrictive (e.g. unweighted), meaning that one cannot simply use the approximate kernel for the more general version. Nevertheless, as we will point out below, our approximate kernel also extends to the unweighted setting (and simple graph setting), with a small loss in parameter.

Maximum k-Vertex Cover

Our first result is a faster FPT-AS for Max k-VC that runs in time $O(1/\varepsilon)^k \cdot \mathsf{poly}(n)$, which improves upon a $(k/\varepsilon)^k \cdot \mathsf{poly}(n)$-time FPT-AS due to Gupta, Lee and Li [29].

▶ **Theorem 1.** *For every $\varepsilon > 0$, there exists an $(1-\varepsilon)$-approximation algorithm for Weighted Max k-VC that runs in time $O(1/\varepsilon)^k \cdot \mathsf{poly}(n)$.*

Perhaps more importantly, our FPT-AS is simple and yields a new insight compared to the previous FPT-ASes [46, 30, 29]. In particular, our algorithm is just the following: restrict ourselves only to the $O(k/\varepsilon)$ vertices with maximum weighted degrees and use brute force to find a k-vertex subset among these vertices that cover edges with maximum total weight.

To demonstrate the differences to the previous algorithms, let us briefly sketch how they work here. The known FPT-ASes [46, 30, 29] all rely on a degree-based argument for the unweighted case due to Marx [46] who consider the following two cases:

[6] For their FPT approximation algorithm [30], the bottleneck is in the reduction from Min k-Cut to Laminar k-Cut which runs in time $2^{O(k^2)} \cdot \mathsf{poly}(n)$. For their $(1+\varepsilon)$-approximation algorithm [29], the bottleneck is in the dynamic programming step which takes $(k/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$ time.

1. The vertex with maximum degree have degree at least k^2/ε. In this case, one can simply take the k vertices with largest degree because the number of edges with both endpoints in the set is at most $\binom{k}{2}$, meaning that it only affects the number of edges covered by at most an ε factor and thus this is already an $(1-\varepsilon)$-approximation for the problem.
2. The vertex with maximum degree have degree at most k^2/ε. The key property in this case is that the number of edges covered by the optimal solution is at most k^3/ε, which is bounded by a function of k. Marx's algorithm then proceeds as follows: (i) guess the number of edges $\ell \leqslant k^3/\varepsilon$ in the optimal solution, (ii) guess (among the k^ℓ possibilities) which vertex (in the solution) that each edge is covered by, (iii) randomly color each edge in the input graph with one of ℓ colors and randomly color each vertex with one of k colors and (iv) finally, determine whether there are k vertices each of different color that covers edges with colors as guessed in Step (ii). Note that Step (iv) can be easily done in polynomial time. Since ℓ is bounded by k^3/ε, the algorithm succeeds with probability at least $k^{-O(k^3/\varepsilon)}$, which can be turned into a randomized algorithm with running time $k^{-O(k^3/\varepsilon)} \cdot \mathsf{poly}(n)$ that succeeds with high probability. Finally, it can be derandomized using standard techniques (see [3]).

The speed-up of Gupta et al. [30, 29] comes from the change in the second case. Roughly speaking, they show that more elaborated coloring techniques can be used, in conjunction with dynamic programming, to speed the second case up to $(k/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$.

Intuitively, our result shows that this case-based analysis is in fact not needed, as it suffices to consider the $O(k/\varepsilon)$ vertices with highest weighted degrees. Moreover, a nice feature about our algorithm is that it works naturally for the weighted case, whereas Gupta et al. needs to employ a discretization argument to deal with this case. (See Section 5.2 in the full version of [30].)

Another feature of our algorithm is that it immediately gives an approximate kernelization for the problem, by restricting to the subgraph induced by the $O(k/\varepsilon)$ vertices and adding self-loops with appropriate weights to compensate the edges from these vertices to the remaining vertices. This results in an $(1-\varepsilon)$-approximate kernelization of $O(k/\varepsilon)$ vertices for Max k-VC:

▶ **Lemma 2.** *For every $\varepsilon > 0$, Weighted Max k-VC admits an $(1-\varepsilon)$-approximate kernelization with $O(k/\varepsilon)$ vertices.*

As stated earlier, the above result is not directly comparable to Lokshtanov et al.'s [43] approximate kernel of $O(k^5/\varepsilon^2)$ vertices for the unweighted version of Max k-VC. Fortunately, our technique also gives an $O(k/\varepsilon^2)$-vertex approximate kernel for the unweighted case, which indeed improves upon Lokshtanov et al.'s result. (See the end of Section 3.2.)

Interestingly, the above approximate kernelization also has an application outside of parameterized complexity: using our approximate kernelization as a preprocessing step, we can directly apply Raghavendra and Tan's SDP-based algorithm for 2SAT with cardinality constraint [52] to give an 0.92-approximation algorithm for Max k-VC in polynomial time. This improves upon the aforementioned polynomial time approximation algorithm of Feige and Langberg [23] which yields $(0.75+\delta)$-approximation for some (small and unspecified) constant $\delta > 0$.

▶ **Corollary 3.** *There exists a polynomial time 0.92-approximation algorithm for Weighted Max k-VC.*

We note here that the approximation guarantee above is even better than the previous best known ratios for some special cases, such as in bipartite graph [4, 13] where the previous best known approximation ratio is 0.821 [13].

Minimum k-Vertex Cover

For the Weighted Min k-VC problem, we give a FPT-AS with similar running time of $O(1/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$ for the problem. Once again, this improves upon the $(k/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$-time algorithm of Gupta et al. [30, 29].

▶ **Theorem 4.** *For every $\varepsilon > 0$, there exists an $(1+\varepsilon)$-approximation algorithm for Weighted Min k-VC that runs in time $O(1/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$.*

We remark that this algorithm is different from the algorithm for Max k-VC and is instead based on a careful branch-and-bound approach. A natural question here is perhaps whether this difference is inherent. While it is unclear how to make this question precise, we provide an evidence that the two problems are indeed of different natures by showing that, in contrast to Max k-VC, a polynomial size approximate kernelization for Min k-VC for any factor less than two is unlikely to exist:

▶ **Lemma 5.** *Assuming the Strong Small Set Expansion Hypothesis (Conjecture 16) and $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, Weighted Min k-VC does not admit a polynomial size $(2-\varepsilon)$-approximate kernelization for any $\varepsilon \in (0, 1]$.*

The above result is under a variant of the Small Set Expansion Hypothesis [50]; please refer to Section 4.2 for the precise definition of the variant. We also note that the above lower bound also applies to the unweighted version; again please see Section 4.2 for more details.

2 Notations

Throughout this note, we think of an edge-weighted graph as a complete graph (self-loops included) where each edge is endowed with a non-negative weight. More specifically, an edge-weighted graph G consists of a vertex set V_G and a weight function $w_G : \binom{V_G}{\leqslant 2} \to \mathbb{R}_{\geqslant 0}$. (Note that, for a set U and a non-negative integer ℓ, we use $\binom{U}{\leqslant \ell}$ and $\binom{U}{\ell}$ to denote the collections of subsets of U of sizes at most ℓ and exactly ℓ respectively.) When the graph is clear from the context, we may drop the subscript G, and we sometimes use w_e to denote $w(e)$ for brevity. For each vertex $v \in V$, we use w-deg(v) to denote its weighted degree, i.e., w-deg$(v) = \sum_{e \in \binom{V_G}{\leqslant 2}, v \in e} w_e$. For a subset $S \subseteq V_G$, we write w-deg(S) to denote $\sum_{v \in S}$ w-deg(v). For subsets $S, T \subseteq V_G$, we use $E_G(S, T)$ to denote the total weight of edges with at least one endpoint in S and at least one endpoint in T; more specifically, $E_G(S, T) = \sum_{e \in \binom{V_G}{\leqslant 2}, e \cap S \neq \emptyset, e \cap T \neq \emptyset} w_G(e)$. Note that $E_G(S, S)$ is the total weight of the edges covered by S; for brevity, we use $E_G(S)$ as a shorthand for $E_G(S, S)$. Finally, we use $\mathrm{OPT}_{\text{Min k-VC}}(G, k)$ and $\mathrm{OPT}_{\text{Max k-VC}}(G, k)$ to denote the optimums of Min k-VC and Max k-VC respectively on the instance (G, k). More formally, $\mathrm{OPT}_{\text{Min k-VC}}(G, k) = \min_{S \in \binom{V_G}{k}} E_G(S)$ and $\mathrm{OPT}_{\text{Max k-VC}}(G, k) = \max_{S \in \binom{V_G}{k}} E_G(S)$.

3 Maximum k-Vertex Cover

We will now prove our results for Max k-VC. To do so, it will be convenient to order the vertices of the input graph V_G based on their weighted degree (ties broken arbitrarily), i.e., let $v_1, \ldots, v_n$ be the ordering of vertices in V_G such that w-deg$_G(v_1) \geqslant \cdots \geqslant \cdot$ w-deg$_G(v_n)$. Moreover, we use V_i to denote the set of i vertices with highest weighted degree, i.e., $V_i = \{v_1, \ldots, v_i\}$.

3.1 A Simple Observation and A Faster FPT-AS

Our main insight to the Weighted Max k-VC problem is that there is always an $(1-\varepsilon)$-approximate solution which is entirely contained in $V_{O(k/\varepsilon)}$, as stated more formally below.

▶ **Observation 6.** *For any $\varepsilon > 0$, let $n' = \min\{k + \lceil k/\varepsilon \rceil, n\}$. Then, there exists $S^* \subseteq V_{n'}$ of size k such that $E_G(S^*) \geqslant (1-\varepsilon) \cdot \mathrm{OPT}_{Max\text{ k-}VC}(G,k)$.*

Note that this implies Theorem 1: we can enumerate all k-vertex subsets of $V_{n'}$ and find an $(1-\varepsilon)$-approximation for Max k-VC in $\binom{|V_{n'}|}{k}\mathsf{poly}(n) = O(n'/k)^k\mathsf{poly}(n) = O(1/\varepsilon)^k\mathsf{poly}(n)$ time.

Before we present a formal proof of the observation, let us briefly give an (informal) intuition behind the proof. Let S_{OPT} be the optimal solution for (G,k). Our goal is to construct another set $S^* \subseteq V_{n'}$ such that $E_G(S^*)$ is roughly the same as $E_G(S_{\mathrm{OPT}})$. To do so, we will just replace each vertex in $S_{\mathrm{OPT}} \setminus V_{n'}$ by a vertex in $V_{n'} \setminus S_{\mathrm{OPT}}$. Intuitively, this should be good for the solution, as we are replacing one vertex with another vertex that has higher weighted degree. However, this argument does not yet work: we might "double count" edges with both endpoints coming from the new vertices. The key point here is that, while we will not be able to avoid this double counting completely, we will be able to pick new vertices such that the total weight of such doubled counted edges is small. This is just because the set $V_{n'}$ is so large that even if we pick a random k vertices from it, the probability that a given added edge is double counted is only $O(\varepsilon)$.

Proof of Observation 6. Note that, if $n' = n$, the statement is obviously true. Hence, we may assume that $n' = k + \lceil k/\varepsilon \rceil$. Let $S_{\mathrm{OPT}} \subseteq V_G$ denote any optimal solution, i.e., any subset of V_G of size k with $E_G(S_{\mathrm{OPT}}) = \mathrm{OPT}_{\text{Max k-VC}}(G,k)$. Let $S^{\mathrm{in}}_{\mathrm{OPT}} = S_{\mathrm{OPT}} \cap V_{n'}$, $S^{\mathrm{out}}_{\mathrm{OPT}} = S_{\mathrm{OPT}} \setminus V_{n'}$ and $U = V_{n'} \setminus S_{\mathrm{OPT}}$.

We construct $S \subseteq V_{n'}$ in randomly as follows. We randomly select a subset $U^* \subseteq U$ of $|S^{\mathrm{out}}_{\mathrm{OPT}}|$ vertices uniformly at random, and let $S = S^{\mathrm{in}}_{\mathrm{OPT}} \cup U^*$. Clearly, S is a subset of $V_{n'}$ of size k. We will show that the expected value of $E_G(S)$ is at least $(1-\varepsilon) \cdot \mathrm{OPT}_{\text{Max k-VC}}(G,k)$. This would imply that there exists $S^* \subseteq V_{n'}$ of size k such that $E_G(S^*) \geqslant (1-\varepsilon) \cdot \mathrm{OPT}_{\text{Max k-VC}}(G,k)$ as desired.

To bound $\mathbb{E}[E_G(S)]$, let us first rearrange $E_G(S)$ as follows.

$$E_G(S) = E_G(S^{\mathrm{in}}_{\mathrm{OPT}}) + E_G(U^*) - E_G(U^*, S^{\mathrm{in}}_{\mathrm{OPT}}). \tag{1}$$

Let $\rho = |S^{\mathrm{out}}_{\mathrm{OPT}}|/|U|$; note here that $\rho \leqslant k/(n'-k) \leqslant \varepsilon$. We can now bound $\mathbb{E}[E_G(U^*, S^{\mathrm{in}}_{\mathrm{OPT}})]$ by

$$\begin{aligned}\mathbb{E}[E_G(U^*, S^{\mathrm{in}}_{\mathrm{OPT}})] &= \sum_{u \in U} \sum_{v \in S^{\mathrm{in}}_{\mathrm{OPT}}} w_{\{u,v\}} \cdot \Pr[u \in U^*] \\ &= \rho \cdot \sum_{u \in U} \sum_{v \in S^{\mathrm{in}}_{\mathrm{OPT}}} w_{\{u,v\}} \leqslant \varepsilon \cdot E_G(S^{\mathrm{in}}_{\mathrm{OPT}})\end{aligned} \tag{2}$$

Moreover, $\mathbb{E}[E_G(U^*)]$ can be rearranged as

$$\begin{aligned}
\mathbb{E}[E_G(U^*)] &= \mathbb{E}\left[\sum_{u\in U^*}\left(\text{w-deg}(u) - \frac{1}{2}\sum_{v\in U^*\setminus\{u\}} w_{\{u,v\}}\right)\right] \\
&= \mathbb{E}\left[\sum_{u\in U}\left(\text{w-deg}(u)\cdot \mathbb{1}[u\in U^*] - \frac{1}{2}\sum_{v\in U\setminus\{u\}} w_{\{u,v\}}\cdot \mathbb{1}[u\in U^*\wedge v\in U^*]\right)\right] \\
&= \sum_{u\in U}\left(\text{w-deg}(u)\cdot \Pr[u\in U^*] - \frac{1}{2}\sum_{v\in U\setminus\{u\}} w_{\{u,v\}}\cdot \Pr[u\in U^*\wedge v\in U^*]\right) \\
&\geqslant \sum_{u\in U}\left(\text{w-deg}(u)\cdot \rho - \frac{1}{2}\sum_{v\in U\setminus\{u\}} w_{\{u,v\}}\cdot \rho^2\right) \\
&\geqslant \rho(1-\rho/2)\cdot\left(\sum_{u\in U}\text{w-deg}(u)\right) \\
&\geqslant \rho(1-\varepsilon)\cdot\left(\sum_{u\in U}\text{w-deg}(u)\right) \qquad (3)
\end{aligned}$$

where in the first inequality we use the fact that $\Pr[u\in U^*\wedge v\in U^*]\leqslant \Pr[u\in U^*]\Pr[v\in U^*]=\rho^2$.

Recall that the vertices are sorted in decreasing order of degrees; thus, for all $u\in U$, we have $\text{w-deg}(u)\geqslant \left(\sum_{v\in S^{\text{out}}_{\text{OPT}}}\text{w-deg}(v)\right)/|S^{\text{out}}_{\text{OPT}}|\geqslant E_G(S^{\text{out}}_{\text{OPT}})/|S^{\text{out}}_{\text{OPT}}|$. From this and (3), we arrive at

$$\mathbb{E}[E_G(U^*)]\geqslant \rho(1-\varepsilon)\cdot|U|\cdot\left(E_G(S^{\text{out}}_{\text{OPT}})/|S^{\text{out}}_{\text{OPT}}|\right) = (1-\varepsilon)\cdot E_G(S^{\text{out}}_{\text{OPT}}) \qquad (4)$$

Plugging (2) and (4) back into (1), we indeed have

$$\mathbb{E}[E_G(S)]\geqslant (1-\varepsilon)(E_G(S^{\text{in}}_{\text{OPT}})+E_G(S^{\text{out}}_{\text{OPT}}))\geqslant (1-\varepsilon)\cdot E_G(S_{\text{OPT}}),$$

which concludes the proof. ◀

3.2 An Approximate Kernel

Observation 6 also naturally gives an $(1-\varepsilon)$-approximate kernel for Weighted Max k-VC where the new instance has $O(k/\varepsilon)$ vertices, as stated below.

Proof of Lemma 2. The reduction algorithm $\mathcal{A}$ works by taking the graph induced on $V_{n'}$ (where $n'=\min\{k+\lceil k/\varepsilon\rceil, n\}$ as in Observation 6) and add appropriate weights to self-loops to compensate for edges going out of $V_{n'}$. More precisely, $\mathcal{A}$ outputs (G',k) where $V_{G'}=V_{n'}$ and $w'_G(\{u,v\})=w'_G(\{u,v\})$ for all $u\neq v\in V_{G'}$ and $w_{G'}(u)=w_G(u)+E_G(\{u\},V_G\setminus V_{n'})$ for all $u\in V_{G'}$.

The solution lifting algorithm $\mathcal{B}$ simply outputs the same solution as its get. It is obvious to see that $E_{G'}(S)=E_G(S)$. Hence, if $E_{G'}(S)\geqslant \alpha\cdot \text{OPT}_{\text{Max k-VC}}(G',k)$, then Observation 6 implies that $E_G(S)=E_{G'}(S)\geqslant \alpha(1-\varepsilon)\cdot\text{OPT}_{\text{Max k-VC}}(G,k)$. This means that $(\mathcal{A},\mathcal{B})$ is an $(1-\varepsilon)$-approximate kernel; moreover, it is obvious that the graph output by $\mathcal{A}$ has size $O(k/\varepsilon)$ as desired. ◀

As mentioned earlier, the above kernel does not directly work for the unweighted case. Let us sketch below how we can modify the above proof to work in this case, albeit with a slightly worse $O(k/\varepsilon^2)$ vertices in the reduced instance. We omit the full proof, which is a simple undergraduate-level exercise, and only describe the main ideas. We do this in two steps; we first modify the proof for weighted graphs without self-loops and then we proceed to unweighted graphs.

- Suppose that the graphs G and G' must not contain any self-loops. Then, instead of adding self-loops as above, $\mathcal{A}$ will add $n_{\text{padded}} = \lceil kn'/\varepsilon \rceil = O(k/\varepsilon^2)$ padded vertices and let the weight between each padded vertex and $u \in V_{n'}$ be $\frac{E_G(\{u\}, V_G \setminus V_{n'})}{n_{\text{padded}}}$. Once again, if we take a look at any set $S \subseteq V_{n'}$, we immediately have $E_G(S) = E_{G'}(S)$. The only additional argument needed is that these padded vertices has little effect on any solution. Indeed, it is simple to see that the weighted degree of each padded vertex is at most $(\varepsilon/k) \cdot \text{OPT}_{\text{Max k-VC}}(G, k)$. Thus, throwing these vertices away from any subset of size k affect the total weights of edges covered by at most $\varepsilon \cdot \text{OPT}_{\text{Max k-VC}}(G, k)$, which implies that this is an $(1 - 2\varepsilon)$-approximate kernel. Adjusting ε appropriately gives the $(1 - \varepsilon)$-approximate kernel with $O(k/\varepsilon^2)$ vertices.
- The above idea naturally adapts to the unweighted case. Instead of adding an edge from every $u \in V_{n'}$ to all the padded vertices, we just add $E_G(\{u\}, V_G \setminus V_{n'})$ edges from each $u \in V_{n'}$ to different padded vertices. These edges are added in a way that each padded vertices has roughly the same degree. It is simple to check that, if the degree of all vertices $u \in V_{n'}$ is at most say k/ε^2, then this works immediately (with the same proof as above). The only issue is when there are vertices with degree larger than k/ε^2. (In this case, the number of edges required to be added may even be larger than n_{padded}!) Nevertheless, this issue can also be easily resolved, by observing that, if any vertex in V_k has degree at least k/ε, then we can always take it in our solution while guaranteeing that the solution still remains within $\varepsilon \cdot \text{OPT}_{\text{Max k-VC}}(G, k)$ of the optimum. Hence, the reduction algorithm can first greedily pick these vertices and then use the padded argument as above; since no large degree vertex remains, the proof of the second step now works and we have the desired approximate kernel.

3.3 Raghavendra-Tan Algorithm and An Improved Approximation

We next describe how our approximate kernel can be used a preprocessing step for the aforementioned algorithm of Raghavendra and Tan [52] for Max 2SAT with cardinality constraint to obtain improved approximation for Weighted Max k-VC.

Recall that the (weighted) Max 2SAT with cardinality constraint is the following problem. Given a collection $\mathcal{C}$ of conjunctions of at most two literals (of variables $\{x_1, \ldots, x_n\}$) and their associated weights, find an assignment to $\{x_1, \ldots, x_n\}$ satisfying $x_1 + \cdots + x_n = k$ that maximizes the total weights of satisfied clauses in $\mathcal{C}$. Raghavendra and Tan [52] device an algorithm with approximation ratio strictly greater than 0.92 for the problem, as stated below.

▶ **Theorem 7** ([52]). *For some $\alpha > 0.92$, there exists an α-approximation algorithm for Max 2SAT with cardinality constraint that runs in time*[7] $n^{\mathsf{poly}(n/k)}$.

[7] The running time of the algorithm is not stated in this form in [52] as they are only concerned about the case where $k = \Omega(n)$, for which the running time is polynomial. To see that the running time is of the form $n^{\mathsf{poly}(n/k)}$, we note that their algorithm needs the variance guaranteed in their Theorem 5.1 to be at most $\mathsf{poly}(k/n)$. This means that they need the SDP solution to be $\mathsf{poly}(k/n)$-independence; to

It is not hard to see that the Weighted Max k-VC can be formulated as Max 2SAT with cardinality constraint: we create a variable x_i for each vertex v_i, and, for each $\{v_i, v_j\} \in \binom{V_G}{\leqslant 2}$, we create a clause $(v_i \vee v_j)$ with weight $w_{\{v_i,v_j\}}$. Obviously, any solution to Max 2SAT satisfying $x_1 + \cdots + x_n = k$ is also a solution of Max k-VC with the same cost. Of course, the only issue in applying Raghavendra and Tan's algorithm here is that its running time $n^{\mathsf{poly}(n/k)}$ is not polynomial when $k = o(n)$. Fortunately, our approximate kernel above precisely circumvents this issue, as the reduction algorithm produces an instance (G', k) where $|V_{G'}| \leqslant O(k/\varepsilon)$. Thus, we can now apply the algorithm and arrives at 0.92 approximation for Weight Max k-VC in polynomial time.

Proof of Corollary 3. Let α be the approximation ratio from Theorem 7 and let $\varepsilon > 0$ be a sufficiently small constant such that $\alpha(1-\varepsilon) \geqslant 0.92$. Let $\mathcal{A}$ be the reduction algorithm for the $(1-\varepsilon)$-approximate kernel as defined in the proof of Lemma 2.

For any instance (G, k) of Weight Max k-VC, we apply $\mathcal{A}$ to arrive at a reduced instance (G', k) where $|V_{G'}| \leqslant O(k/\varepsilon)$. We then formulate the instance (G', k) as an instance of Max 2SAT with cardinality constraint and apply the Raghavendra-Tan algorithm, which gives an α-approximate solution, i.e., a set $S \subseteq V_{G'}$ of size k such that $E_{G'}(S) \geqslant \alpha \cdot \mathrm{OPT}_{\text{Max k-VC}}(G', k) \geqslant \alpha(1-\varepsilon) \cdot \mathrm{OPT}_{\text{Max k-VC}}(G, k) \geqslant 0.92 \cdot \mathrm{OPT}_{\text{Max k-VC}}(G, k)$. Note that the Raghavendra-Tan algorithm runs in $k^{\mathsf{poly}(|V_{G'}|/k)} = k^{\mathsf{poly}(1/\varepsilon)}$ time. Hence, we have found a 0.92-approximate solution for (G, k) in polynomial time. ◀

4 Minimum k-Vertex Cover

4.1 A Faster FPT-AS

We now present our result on Weighted Min k-VC, starting with the faster FPT-AS (Theorem 4). It will be more convenient for us to work with a multicolored version of the problem, which we call *Multicolored Min k-VC*. In Multicolored Min k-VC, we are given G, k as before and also a coloring $\chi : V_G \to [k]$. A set $S \subseteq V_G$ is said to be *colorful* if every vertex in S is assigned a different color, i.e., $|\chi(S)| = |S|$. The goal of Multicolored Min k-VC is to find a colorful $S \subseteq V_G$ of size k that maximizes $E_G(S)$. We overload the notation $\mathrm{OPT}_{\text{Min k-VC}}$ and also use it to denote the optimum of Multicolored Min k-VC; that is, we let $\mathrm{OPT}_{\text{Min k-VC}}(G, k, \chi) = \min_{S \in \binom{V_G}{k}, |\chi(S)|=k} E_G(S)$.

The main theorem of this section is the following FPT-AS for Multicolored Min k-VC.

▶ **Theorem 8.** *For any $\varepsilon > 0$, there exists an $(1+\varepsilon)$-approximation algorithm for Multicolored Min k-VC that runs in time $O(1/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$.*

We note here that the above lemma immediately gives an FPT-AS for (uncolored) Weight Min k-VC with similar running time (i.e. Theorem 4) via standard color-coding technique [3]. Specifically, they show how to construct a family $\mathcal{F}$ of k-perfect hash functions from $V_G \to \{1, \ldots, k\}$ in $2^{O(k)} \cdot \mathsf{poly}(n)$ time. By running the FPT-AS from Theorem 8 on (G, k, χ) for all $\chi \in \mathcal{F}$ and take the best solution among the outputs, we arrive at the FPT-AS for (uncolored) Weight Min k-VC.

We now proceed to discuss the intuition behind Theorem 8. The algorithm consists of two parts: subgraph generation and dynamic programming. Roughly speaking, the subgraph generation part will, for each set of colors $C \subseteq [k]$, generate connected colorful subsets

find such a solution, the running time required is $n^{\mathsf{poly}(n/k)}$ (see Theorem 4.1 in that paper).

$T \subseteq V_G$ whose color is C and record the minimum $E_G(T)$ in the table cell DP[C]. The second part of the algorithm then uses a standard dynamic programming to find a colorful k-vertex S with minimum $E_G(S)$.

For the purpose of exposition, let us assume for the moment that our graph is unweighted. The subgraph generation part is the heart of the algorithm, and, if not implemented in a careful manner, will be too slow. For instance, the trivial implementation of this is as a recursive function that maintains a set of included vertices S_{INCLUDED} and a set of active vertices S_{ACTIVE}. This function then picks any vertex $u \in S_{\text{ACTIVE}}$ and tries to select at most k neighbors of u to add into S_{INCLUDED} and S_{ACTIVE}; the function then remove u from S_{ACTIVE} and recursively call itself on this new sets. (Note that in this step it also makes sure that the set S_{INCLUDED} remains colorful; otherwise, the recursive call is not made.) The function stops when S_{ACTIVE} is empty and update DP[C] to be the minimum between the current value and $E_G(S_{\text{INCLUDED}})$. As the reader may have already noticed, while this algorithm records (exactly) the correct answer into the table, it is very slow. In particular, if say we run this on a complete graph, then it will generates $n^{\Theta(k)}$ subgraphs.

The algorithm of Gupta, Lee and Li [30, 29], while not stated in this exact form, can be viewed as a more careful implementation of this approach. In particular, they use the observation of Marx [46] (that was also outlined outline in Section 1.1): for unweighted graphs, if the optimal solution has any vertex with degree at least $\binom{k}{2}/\varepsilon$, simply picking the k vertices with minimum degrees would already be an $(1+\varepsilon)$-approximate solution. In other words, one may assume that the graph has degree bounded by $\binom{k}{2}/\varepsilon = O(k^2/\varepsilon)$. When this is the case, the algorithm from the previous paragraph in fact runs in $O(k/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$ time; the reason is that the number of choices to be made when adding a vertex is only $O(k^2/\varepsilon)$ instead of n as before. Hence, the running time becomes $O(k^2/\varepsilon)^k \cdot \mathsf{poly}(n) = (k/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$.

To obtain further speed up, we observe that, if at most $\varepsilon/2$ fraction of neighbors of a vertex u lies in the optimal solution, then ignoring all of them completely while branching would change the number of covered edges by factor of no more than ε. (This is shown formally in the proof below.) In other words, instead of trying all subsets of at most k neighbors of u. We may only try subsets with at least $d\varepsilon/2$ (and at most k) neighbors of u where d is the degree of u. The point here is that, while there are still $\exp(d)$ branches, we are adding at least $d\varepsilon/2$ vertices. Hence, the "branching factor per vertex added" is small: namely, for $j \geqslant d\varepsilon/2$, the "branching factor per vertex added" is only $\binom{d}{j}^{1/j} \leqslant ed/j \leqslant O(1/\varepsilon)$. This indeed gives the running time of $O(1/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$. (Note that such branching may result in a connected component being separated; however, when this is the case, the number of edges between the generated parts must be small anyway.)

Let us now shift our discussion to the edge-weighted graph case. Once again, as we will show formally in the proof, throwing away the edges adjacent to u with total weight at most $(\varepsilon/2) \cdot \text{w-deg}(u)$ only affects the solution value by no more than ε factor. However, this observation alone is not enough; specifically, unlike the unweighted case, this does not guarantee that many vertices must be selected. As an example, if there is a vertex v where $w_{\{u,v\}} = 0.5 \cdot \text{w-deg}(u)$, then even the set $\{v\}$ should be consider when we branch. Nevertheless, it is once again possible to show that, we can select a collection $\mathcal{T}$ of representative subsets such that, for any set $S \subseteq V_G$ (the true optimal set), we can arrive in a subset in $\mathcal{T}$ by throwing away vertices whose edges to u are of total weight at most $(\varepsilon/2) \cdot \text{w-deg}(u)$. In other words, it is "safe" to just consider branching with subsets in $\mathcal{T}$ instead of all subsets. Again, the collection $\mathcal{T}$ will satisfy the property that the "branching factor per vertex added" is small; that is, for any j, the number of j-element subsets that belong to $\mathcal{T}$ is at most $O(1/\varepsilon)^j$. The existence and efficient construction of such $\mathcal{T}$ is stated below in a more general form.

Note that, in the context of subgraph generation algorithm, one should think of $\delta = \varepsilon/2$, $\ell = n - 1$ (all vertices except u itself) and $P = \text{w-deg}(u) - w_{\{u\}}$.

▶ **Lemma 9.** *Let $a_1, \ldots, a_\ell \geqslant 0$ be any non-negative real numbers, let $\delta > 0$ be any positive real number, and let $P = \sum_{i \in [\ell]} a_i$. Then, there exists a collection $\mathcal{T}$ of subsets of $[\ell]$ such that*

- **(i)** *For all $j \in [\ell]$, we have $\left|\mathcal{T} \cap \binom{[\ell]}{j}\right| \leqslant O(1/\delta)^j$, and,*
- **(ii)** *For any $S \subseteq [\ell]$, there exists $T \in \mathcal{T}$ such that $T \subseteq S$ and $\sum_{i \in (S \setminus T)} a_i \leqslant \delta \cdot P$.*

Moreover, for any $j \in [\ell]$, $\mathcal{T} \cap \binom{[\ell]}{\leqslant j}$ can be computed in $O(1/\delta)^{O(j)} \ell^{O(1)}$ time.

Proof. Let $\pi : [\ell] \to [\ell]$ be any permutation such that $a_{\pi(1)} \geqslant \cdots \geqslant a_{\pi(\ell)}$. For each $j \in [\ell]$, we construct $\mathcal{T} \cap \binom{[\ell]}{j}$ by taking all j-element subsets of $\{\pi(1), \ldots, \pi(\min\{j \cdot \lceil 1/\delta \rceil, \ell\})\}$. We have

$$\left|\mathcal{T} \cap \binom{[\ell]}{j}\right| \leqslant \binom{j \cdot \lceil 1/\delta \rceil}{j} \leqslant \left(\frac{ej \cdot \lceil 1/\delta \rceil}{j}\right)^j \leqslant O(1/\delta)^j.$$

Moreover, it is clear that the set $\mathcal{T} \cap \binom{[\ell]}{j}$ can be generated in time polynomial in the size of the set and ℓ, which is $O(1/\delta)^{O(j)} \ell^{O(1)}$ as desired.

Finally, we will prove ii. Consider any subset $S \subseteq [\ell]$ and suppose that its elements are $\pi(i_1), \ldots, \pi(i_m)$. We pick the set T as follows: let t be the largest index such that $i_t \leqslant t \cdot \lceil 1/\delta \rceil$ and let $T = \{\pi(i_1), \ldots, \pi(i_t)\}$. Since $i_t \leqslant t \cdot \lceil 1/\delta \rceil$, T is a t-element subset from $\{\pi(1), \ldots, \pi(\min\{t \cdot \lceil 1/\delta \rceil, \ell\})$ and hence T belongs to $\mathcal{T}$. To prove ii, observe that, by definition of t, we have $i_g > g \cdot \lceil 1/\delta \rceil$ for all $g > t$. This means that

$$\sum_{i \in (S \setminus T)} a_i = \sum_{g=t+1}^{m} a_{\pi(g)} \leqslant \sum_{g=t+1}^{m} \left(\frac{1}{\lceil 1/\delta \rceil} \sum_{i=(g-1)\cdot\lceil 1/\delta \rceil + 1}^{g \cdot \lceil 1/\delta \rceil} a_i \right) \leqslant \frac{1}{\lceil 1/\delta \rceil} \sum_{i \in [\ell]} a_i \leqslant \delta \cdot P,$$

which concludes the proof. ◀

With the above lemma ready, we proceed to the proof of Theorem 8.

Proof of Theorem 8. The proof is based on the ideas outlined above. For simplicity, we will describe the algorithm that computes an approximation for $\text{OPT}_{\text{Min k-VC}}(G, k, \chi)$ rather than a subset $S \subseteq V_G$, i.e., it will output a number between $\text{OPT}_{\text{Min k-VC}}(G, k, \chi)$ and $(1 + \varepsilon) \cdot \text{OPT}_{\text{Min k-VC}}(G, k, \chi)$. It is not hard to see that the algorithm can be turned to provide a desired set as well.

As stated above, the algorithm consists of two parts: the subgraph generation part, and the dynamic programming part. The subgraph generation algorithm, which is shown below as Algorithm 1, is very much the same as stated earlier: it takes as an input the sets S_{ACTIVE} and S_{INCLUDED} (in addition to (G, k, χ)). If there is no more active vertex in S_{ACTIVE}, then it just updates the table DP to reflect $E_G(S_{\text{INCLUDED}})$. Otherwise, it pick a vertex u and try to branch on every representative T from $\mathcal{T}$ from Lemma 9 where the $\{a_i\}$'s are defined as $a_v = w_{\{u,v\}}$ for all $v \neq \{u\}$ and $\delta = \varepsilon/2$.

The dynamic programming (main algorithm) proceeds in a rather straightforward manner: after initializing the table, the main algorithm calls the subgraph generation subroutine starting with each vertex. Then, it uses dynamic programming to updates the table DP to reflect the fact that the answer may consist of many connected components. Finally, it outputs $\text{DP}[\{1, \ldots, k\}]$. The pseudo-code for this is given below as Algorithm 2.

Algorithm 1

1: **procedure** SubgraphGen($G, k, \chi, S_{\text{active}}, S_{\text{included}}$)
2: **if** $S_{\text{active}} = \emptyset$ **then**
3: $\text{DP}[\chi(S_{\text{included}})] \leftarrow \min\{\text{DP}[\chi(S_{\text{included}})], E_G(S_{\text{included}})\}$
4: **else**
5: $u \leftarrow$ Any element of S_{active}
6: $S_{\text{active}} \leftarrow S_{\text{active}} \setminus \{u\}$
7: $\mathcal{T} \leftarrow$ Subsets generated by Lemma 9 for $a_v = w_{\{u,v\}}$ for all $v \neq u$ and $\delta = \varepsilon/2$.
8: **for** $T \subseteq \mathcal{T} \cap \binom{V_G \setminus \{u\}}{\leqslant k}$ **do**
9: **if** $T \cap S_{\text{included}} = \emptyset$ **and** $S_{\text{included}} \cup T$ is colorful **then**
10: SubgraphGen($G, k, \chi, S_{\text{active}} \cup T, S_{\text{included}} \cup T$)
11: **end procedure**

Algorithm 2

1: **procedure** Min_k-VC(G, k, χ)
2: **for** $C \subseteq [k]$ **do**
3: $\text{DP}[C] \leftarrow \infty$
4: **for** $u \in V_G$ **do**
5: SubgraphGen($G, k, \chi, \{u\}, \{u\}$)
6: **for** $C \subseteq [k]$ in increasing order of $|C|$ **do**
7: **for** $C' \subseteq C$ **do**
8: $\text{DP}[C] \leftarrow \min\{\text{DP}[C], \text{DP}[C'] + \text{DP}[C \setminus C']\}$
9: **return** $\text{DP}[[k]]$
10: **end procedure**

Running Time Analysis. We will show that the running time of the algorithm is indeed $O(1/\varepsilon)^{O(k)}$. It is obvious that the dynamic programming step takes only $2^{O(k)} \cdot \mathsf{poly}(n)$ time, and it is not hard to see that each call to SubgraphGen, without taking into account the time spent in the recursed calls (Step 10), takes only $O(1/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$ time (because the bottleneck is the generation of $\mathcal{T} \cap \binom{V_G \setminus \{u\}}{\leqslant k}$ and this takes only $O(1/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$ time as guaranteed by Lemma 9). Thus, it suffices for us to show that, for each $u \in V$, SubgraphGen($G, k, \chi, \{u\}, \{u\}$) only generates $O(1/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$ leaves in the recursion tree. (By *leaves*, we refer to calls SubgraphGen($G, k, \chi, S_{\text{active}}, S_{\text{included}}$) where $S_{\text{active}} = \emptyset$. Note that, if SubgraphGen($G, k, \chi, \emptyset, S_{\text{included}}$) is called multiple times for the same S_{included}, we count each call separately.) The proof is a formalization of the "branching factor per vertex added" idea outlined before the proof.

In fact, we will prove a more general statement: for all colorful subsets $S_{\text{active}} \subseteq S_{\text{included}}$, SubgraphGen($G, k, \chi, S_{\text{active}}, S_{\text{included}}$) results in at most $(C/\varepsilon)^{2k - |S_{\text{included}}| - |S_{\text{included}} \setminus S_{\text{active}}|}$ leaves for some $C > 0$. In particular, let $C' > 0$ be a constant such that Lemma 9 gives the bound $|\mathcal{T} \cap \binom{[\ell]}{j}| \leqslant (C'/\delta)^j$; we will prove the statement for $C = 2C' + 2$.

We prove by induction on decreasing order of $|S_{\text{included}}|$ and $|S_{\text{included}} \setminus S_{\text{active}}|$ respectively. In the base case where $|S_{\text{included}}| = k$, the statement is obviously true, since the condition in Line 9 ensures that no more subroutine is executed. In another base case where $|S_{\text{included}} \setminus S_{\text{active}}| = |S_{\text{included}}|$, the statement is also obviously true since, in this case, we simply have $S_{\text{active}} = \emptyset$.

For the inductive step, suppose that, for some $0 \leqslant i < k$ and $1 \leqslant j \leqslant i$, the statement holds for all colorful subsets $S_{\text{ACTIVE}} \subseteq S_{\text{INCLUDED}}$such that $|S_{\text{INCLUDED}}| > i$, or, $|S_{\text{INCLUDED}}| = i$ and $|S_{\text{ACTIVE}}| < j$. Now, consider any colorful subsets $S_{\text{ACTIVE}} \subseteq S_{\text{INCLUDED}}$ such that $|S_{\text{INCLUDED}}| = i$ and $|S_{\text{ACTIVE}}| = j$. We will argue below that SubgraphGen$(G, k, \chi, S_{\text{ACTIVE}}, S_{\text{INCLUDED}})$ results in at most $(C'/\varepsilon)^{2k-i-(i-j)}$ leaves.

To do so, first observe that (1) in every recursive call, $|S_{\text{INCLUDED}} \setminus S_{\text{ACTIVE}}|$ increases by one (namely u becomes inactive) and (2) for every $0 \leqslant t \leqslant k-i$, the number of recursive calls for which $|S_{\text{INCLUDED}}|$ increases by t is at most $|\mathcal{T} \cap \binom{V_G \setminus \{u\}}{t}| \leqslant (C'/\varepsilon)^t$. As a result, by the inductive hypothesis, the number of leaves generated by SubgraphGen$(G, k, \chi, S_{\text{ACTIVE}}, S_{\text{INCLUDED}})$ is at most

$$\begin{aligned}\sum_{t=0}^{k-i}(C'/\varepsilon)^t \cdot (C/\varepsilon)^{2k-(i+t)-(i-j+1)} &= (C/\varepsilon)^{2k-i-(i-j+1)} \cdot \left(\sum_{t=0}^{k-i}(C'/C)^t\right) \\ \text{(Since } C \geqslant 2C') &\leqslant (C/\varepsilon)^{2k-i-(i-j+1)} \cdot 2 \\ \text{(Since } C \geqslant 2) &\leqslant (C/\varepsilon)^{2k-i-(i-j)}\end{aligned}$$

as desired.

In conclusion, for all colorful $S_{\text{ACTIVE}} \subseteq S_{\text{INCLUDED}}$, SubgraphGen$(G, k, \chi, S_{\text{ACTIVE}}, S_{\text{INCLUDED}})$ generates at most $(C/\varepsilon)^{2k-|S_{\text{INCLUDED}}|-|S_{\text{INCLUDED}} \setminus S_{\text{ACTIVE}}|}$ leaves. As argued above, this implies that the running time of the algorithm is at most $O(1/\varepsilon)^{O(k)} \cdot \mathsf{poly}(n)$.

Approximation Guarantee Analysis. We will now show that the output lies between $\text{OPT}_{\text{Min } k\text{-VC}}(G, k, \chi)$ and $(1+\varepsilon) \cdot \text{OPT}_{\text{Min } k\text{-VC}}(G, k, \chi)$. For convenience, let us define DP* to be the value of table DP filled by SubgraphGen calls; that is, this is the table before Line 6 in Algorithm 2. Observe the following relationship between DP and DP*:

$$\text{DP}[C] = \min_{\text{Partition } P \text{ of } C} \sum_{C' \in P} \text{DP}^*[C']. \tag{5}$$

It is now rather simple to see that the output is at least $\text{OPT}_{\text{Min } k\text{-VC}}(G, k, \chi)$. To do so, observe that, for any $C \subseteq [k]$, DP$^*[C]$ is equal $E_G(S_C)$ for some colorful $S_C \subseteq V_G$ with $\chi(S_C) = C$. This, together with (5), implies that the output must be equal to $\sum_{C' \in P} E_G(S_{C'})$ for some partition P of $[k]$ and colorful $S_{C'}$'s such that $\chi(S_{C'}) = C'$. Observe that this value is at least $E_G\left(\bigcup_{C' \in P} S_{C'}\right)$, which is at least $\text{OPT}_{\text{Min } k\text{-VC}}(G, k, \chi)$ since $\bigcup_{C' \in P} S_{C'}$ is a colorful set of size k.

Next, we will show that the output (i.e. DP$[[k]]$) is at most $(1+\varepsilon) \cdot \text{OPT}_{\text{Min } k\text{-VC}}(G, k, \chi)$. The following proposition is at the heart of this proof:

▶ **Proposition 10.** *For any non-empty colorful subset $S \subseteq V_G$, there exists a non-empty $S^{rep} \subseteq S$ such that*

$$DP^*[\chi(S^{rep})] \leqslant E_G(S^{rep}) \text{ and } E_G(S^{rep}, S \setminus S^{rep}) \leqslant \delta \cdot \text{w-deg}(S^{rep}).$$

Proof of Proposition 10. Let v be any vertex in S. Let us consider the call SubgraphGen$(G,\ \chi, k, \{v\}, \{v\})$. Consider traversing the following single branch in every execution of Step 10: pick $T \in \mathcal{T}$ such that $T \subseteq (S \setminus S_{\text{INCLUDED}})$ and $\sum_{i \in (S \setminus S_{\text{INCLUDED}}) \setminus T} w_{\{u,i\}} \leqslant \delta \cdot \sum_{i \in V_G} w_{\{u,i\}} = \delta \cdot \text{w-deg}_G(u)$. (We remark that such T is guaranteed to exist by Lemma 9; if there are more than one such T's, just choose an arbitrary one.) Suppose that always choosing such branch ends in a call SubgraphGen$(G, k, \chi, \emptyset, S^{\text{rep}})$. We will show that S^{rep} satisfies the desired properties.

First of all, observe that the fact we always choose $T \subseteq S$ ensures that $S^{\text{rep}} \subseteq S$ and that, since SubgraphGen$(G, k, \chi, \emptyset, S^{\text{rep}})$ is executed, we indeed have $\text{DP}[\chi(S^{\text{rep}})] \leqslant E_G(S^{\text{rep}})$. Hence, we are only left to argue that $E_G(S^{\text{rep}}, S \setminus S^{\text{rep}}) \leqslant \delta \cdot \text{w-deg}(S^{\text{rep}})$. To see that this is the case, observe that the second property of the T's chosen implies that $\sum_{i \in S \setminus S^{\text{rep}}} w_{\{u,i\}} \leqslant \delta \cdot \text{w-deg}(u)$. Summing this inequality over all $u \in S^{\text{rep}}$ immediately yields $E_G(S^{\text{rep}}, S \setminus S^{\text{rep}}) \leqslant \delta \cdot \text{w-deg}(S^{\text{rep}})$. ◀

With Proposition 10 ready, we can now prove that $\text{DP}[[k]] \leqslant (1+\varepsilon) \cdot \text{OPT}_{\text{Min } k\text{-VC}}(G, k, \chi)$. Let $S_{\text{OPT}} \subseteq V_G$ denote an optimal solution to the problem, i.e., S_{OPT} is a colorful k-vertex subset such that $E_G(S_{\text{OPT}}) = \text{OPT}_{\text{Min k-VC}}(G, k, \chi)$. Let $S_1 = S_{\text{OPT}}$. For $i = 1, \ldots,$ if $S_i \neq \emptyset$, we apply Proposition 10 to find a non-empty subset $S_i^{\text{rep}} \subseteq S_i$ such that

$$\text{DP}^*[\chi(S_i^{\text{rep}})] \leqslant E_G(S_i^{\text{rep}}) \text{ and } E_G(S_i^{\text{rep}}, S_{i+1}) \leqslant \delta \cdot \text{w-deg}(S_i^{\text{rep}}). \tag{6}$$

where $S_{i+1} = S_i \setminus S_i^{\text{rep}}$.

Observe here that $\{S_i^{\text{rep}}\}_{i \geqslant 1}$ is a partition of S_{OPT}. Thus, from (5) and (6), we have

$$\text{DP}[[k]] \overset{(5)}{\leqslant} \sum_{i \geqslant 1} \text{DP}^*[\chi(S_i^{\text{rep}})] \overset{(6)}{\leqslant} \sum_{i \geqslant 1} E_G(S_i^{\text{rep}}). \tag{7}$$

On the other hand, observe that $E_G(S_i) = E_G(S_i^{\text{rep}}) + E_G(S_{i+1}) - E_G(S_i^{\text{rep}}, S_{i+1})$. Thus, we have

$$\begin{aligned} E_G(S_{\text{OPT}}) &= \sum_{i \geqslant 1} (E_G(S_i) - E_G(S_{i+1})) \\ &= \sum_{i \geqslant 1} E_G(S_i^{\text{rep}}) - \sum_{i \geqslant 1} E_G(S_i^{\text{rep}}, S_{i+1}) \\ &\overset{(6)}{\geqslant} \sum_{i \geqslant 1} E_G(S_i^{\text{rep}}) - \delta \cdot \sum_{i \geqslant 1} \text{w-deg}(S_i^{\text{rep}}) \\ &= \sum_{i \geqslant 1} E_G(S_i^{\text{rep}}) - \delta \cdot \text{w-deg}(S_{\text{OPT}}). \end{aligned} \tag{8}$$

Finally, from (7), (8) and $\text{w-deg}(S_{\text{OPT}}) \leqslant 2 \cdot E_G(S_{\text{OPT}})$, we have $\text{DP}[[k]] \leqslant (1 + 2\delta) \cdot E_G(S_{\text{OPT}}) = (1 + \varepsilon) \cdot \text{OPT}_{\text{Min } k\text{-VC}}(G, k, \chi)$ which concludes the proof. ◀

4.2 Non-Existence of Polynomial Size Approximate Kernel

The above FPT-AS and the equivalence between existence of FPT approximation algorithm and approximate kernel [43] immediately implies that there exists an $(1-\varepsilon)$-approximate kernel for Weighted Min k-VC. However, this naive approach results in an approximate kernel of size $O(1/\varepsilon)^{O(k)}$. A natural question is whether there exists a polynomial-size approximate kernel for Weighted Min k-VC (similar to Weighted Max k-VC). In this section, we show that the answer to this question is likely a negative, assuming a variant of the Small Set Expansion Conjecture.

Our proof follows the framework of Lokshtanov et al. [43]. Let us recall that an equivalence relation R over strings on a finite alphabet Σ is said to be *polynomial* if (i) whether $x \sim y$ can be checked in $\mathsf{poly}(|x|+|y|)$ time and (ii) for every $n \in \mathbb{N}$, Σ^n has at most $\mathsf{poly}(n)$ equivalence classes. The framework of Lokshtanov et al. uses the notion of α-gap cross composition, as defined below. (This is based on the cross composition in the exact settings from [12].)

▶ **Definition 11** (α-gap cross composition [43])**.** Let L be a language and Π be a parameterized minimization problem. We say that L *α-gap cross composes* into Π (for $\alpha \leqslant 1$), if there is a polynomial equivalence relation R and an algorithm which, given strings $x_1, \cdots, x_t$ from the same equivalence class of R, computes an instance (y, k) of Π and $r \in \mathbb{R}$, in time $\mathsf{poly}(\sum_{i=1}^{t} |x_i|)$ such that the following holds:

(i) (Completeness) $\mathrm{OPT}_\Pi(y, k) \leqslant r$ if $x_i \in L$ for some $1 \leqslant i \leqslant t$,
(ii) (Soundness) $\mathrm{OPT}_\Pi(y, k) > r\alpha$ if $x_i \notin L$ for all $i \in [t]$, and,
(iii) k is bounded by a polynomial in $\log t + \max_{1 \leqslant i \leqslant t} |x_i|$.

A parameterized optimization problem is said to be *nice* if, given a solution to the problem, its cost can be computed in polynomial time. (Clearly, Weighted Min k-VC is nice.) The main tool from [43] is that any problem that α-gap cross composes to a nice parameterized optimization problem Π must be in $\mathsf{coNP/poly}$ if Π has α-approximate kernel[8]. In other words, if an NP-hard language α-gap cross composes to Π, then Π does not have α-approximate kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

▶ **Lemma 12** ([43])**.** *Let L be a language and Π be a nice parameterized optimization problem. If L α-gap cross composes to Π, and Π has a polynomial size α-approximate kernel, then $L \in \mathsf{coNP/poly}$.*

As stated earlier, our lower bound will be based on the Small Set Expansion Hypothesis (SSEH) [50]. To state the hypothesis, let us first recall the definition of edge expansion; for a graph G, the *edge expansion* of a subset of vertices $S \subseteq V_G$ is defined as $\Phi(S) := \frac{E_G(S, V_G \setminus S)}{\text{w-deg}(S)}$. Roughly speaking, SSEH, which was proposed in [50], states that it is NP-hard to determine whether (completeness) there is a subset of a specified size with very small edge expansion or (soundness) every subset of a specified size has edge expansion close to one. This is formalized below.

▶ **Definition 13** ($\mathsf{SSE}(\delta, \eta)$)**.** Given an unweighted regular graph G, distinguish between:

- (Completeness) There exists $S \subseteq V_G$ of size $\delta|V_G|$ such that $\Phi(S) \leqslant \eta$.
- (Soundness) For every $S \subseteq V_G$ of size $\delta|V_G|$, $\Phi(S) > 1 - \eta$.

▶ **Conjecture 14** (Small Set Expansion Hypothesis [50])**.** *For every $\eta > 0$, there exists $\delta = \delta(\eta) > 0$ such that SSE(δ, η) is NP-hard.*

Before we state the variant of SSEH that we will use, let us demonstrate why we need to strengthen the hypothesis. To do so, let us consider the $(2 - \varepsilon)$-factor hardness of approximation of Min k-VC as proved in [25], which our construction will be based on. The reduction takes in an input G to SSE(δ, η) and simply just outputs (G, k) where $k = \delta|V_G|$. The point is that, in a d-regular graph, a set S covers exactly $d(1 + \Phi(S))|S|/2$ edges. This means that, in the completeness case, there is a set S of size k that covers only $d(1 + \eta)k/2$ edges, whereas, in the soundness case, any set S of size k covers at least $d(2 - \eta)k/2$ edges. By selecting η sufficiently small, the ratio between the two cases is at least $(2 - \varepsilon)$, and hence [25] arrives at their $(2 - \varepsilon)$-factor inapproximability result.

Now, our cross composition is similar to this, except that we need to be to handle multiple instances at once. More specifically, given instance $G_1, \ldots, G_t$ of SSE(δ, η) where all $G_1, \ldots, G_t$ are d-regular for some d and $|V_{G_1}| = \cdots = |V_{G_t}|$, we want to produce an instance (G^*, k) where G^* is the disjoint union of $G_1, \ldots, G_t$ and $k = \delta|V|$. Once again, the

[8] We note that the result of [43] works even with a weaker notion than α-approximate kernel called *α-approximate compression*; see Definition 5.5 and Theorem 5.9 of [43] for more details.

completeness case works exactly as before. The issue lies in the soundness case: even though we know that every $S_i \subseteq V_{G_i}$ of size k has expansion close to one, it is possible that there exists $S_i \subseteq V_{G_i}$ of size much smaller than k that has small expansion. For instance, it might even be that $G_1, \ldots, G_t$ each contains a connected component of size k/t. In this case, we can take the union of these components and arrive at a set of size k that covers $dk/2$ edges, which is even smaller than the completeness case! In other words, for the composition to work, we want the soundness of SSEH to consider not only S's of size k, but also S's of size *at most* k. With this in mind, we can formalize our strengthened hypothesis as follows.

▶ **Definition 15** (Strong-SSE(δ, η))**.** Given an unweighted regular graph G, distinguish between:

- (Completeness) There exists $S \subseteq V_G$ of size $\delta|V_G|$ such that $\Phi(S) \leqslant \eta$.
- (Soundness) For every $S \subseteq V_G$ of size at most $\delta|V_G|$, $\Phi(S) > 1 - \eta$.

▶ **Conjecture 16** (Strong Small Set Expansion Hypothesis)**.** *For every $\eta > 0$, there exists $\delta = \delta(\eta) > 0$ such that Strong-SSE(δ, η) is NP-hard.*

We remark that it is known that a strengthening of SSEH where the soundness case is required for all S of size in $[\beta\delta|V|, \delta|V|]$ for any $\beta > 0$ is known to be equivalent to the original SSEH. (See Appendix A.2 of the full version of [51] for a simple proof.) This is closely related to what we want above, except that we need this to holds even for $|S| = o(|V|)$. To the best of our knowledge, the Strong SSEH as stated above is not known to be equivalent to the original SSEH.

Proof of Lemma 5. Let ε be any number that lies in $(0, 1]$. Let η be $\varepsilon/2$, and let $\delta = \delta(\eta) > 0$ be as guaranteed by Conjecture 14. We will show that Strong-SSE(δ, η) $(2 - \varepsilon)$-gap cross composes[9] into Min k-VC, which together with Lemma 12 immediately implies the statement in the lemma.

We define an equivalence relation R on instances of Strong-SSE(δ, η) by $G \sim G'$ iff $|V_G| = |V_{G'}|$ and w-deg(G) = w-deg(G'). It is obvious that R is polynomial. Given t instances $G_1, \ldots, G_t$ from the same equivalence class of R where $n = |V_{G_1}| = \cdots = |V_{G_t}|$ and $d =$ w-deg$(G_1) = \cdots =$ w-deg(G_t), we create an instance (G^*, k) of Min k-VC by letting G^* be the (disjoint) union of $G_1, \ldots, G_t$, $k = \delta n$, and $r = d\delta n(1 + \eta)/2$. We next argue the completeness and soundness of the composition.

Completeness. Suppose that, for some $i \in [t]$, there exists $S \subseteq V_{G_i}$ of size δn such that $\Phi_{G_i}(S) \leqslant \eta$. Then, the number of edges covered by S (in both G_i and G^*) is $d\delta n(1 + \Phi(S))/2 \leqslant d\delta n(1 + \eta)/2$. In other words, $\mathrm{OPT}_{\text{Max } k\text{-VC}}(G^*, k) \leqslant r$ as desired.

Soundness. Suppose that, for all $i \in [t]$ and $S \subseteq V_{G_i}$ of size at most δn, we have $\Phi_{G_i}(S) > (1 - \eta)$. Consider any set $S^* \subseteq V_{G^*}$ of size δn. Let S_i denote $S^* \cap V_{G_i}$. Observe that the number of edges covered by S^* is

$$\sum_{i \in [t]} d|S_i|(1 + \Phi_{G_i}(S_i))/2 \geqslant \sum_{i \in [t]} d|S_i|(2 - \eta)/2 = d\delta n(2 - \eta)/2 \geqslant (2 - \varepsilon)r,$$

[9] Note that strictly speaking Strong-SSE(δ, η) is not a language, but rather a promise problem (cf. [27]). Nevertheless, the notion of gap cross composes extends naturally to promise problems; the only changes are that in the yes case $x_i \in L$ should be changed to $x_i \in L_{\text{YES}}$ and in the no case $x_i \notin L$ should be changed to $x_i \in L_{\text{NO}}$. The result in Lemma 12 also holds for this case; for instance, see Lemma 5.11 and Theorem 5.12 of [43], where the gap cross composition also starts from a promise problem (Gap-Longest-Path).

where the first inequality comes from our assumption and the second comes from our choice of η. Thus, we have $\mathrm{OPT}_{\text{Max } k\text{-VC}}(G^*, k) > (2-\varepsilon)r$ as desired. ◀

We note here that the above proof produces G^* that is unweighted. As a result, the lower bound also applies for Unweighted Min k-VC.

5 Concluding Remarks

Let us make a few brief remarks regarding the tightness of running times of our algorithms.

- The W[1]-hardness proofs of Max k-VC and Min k-VC in [28] also implies that, even in the unweighted case, if we can approximate the problems to within $(1-1/n^2)$ and $(1+1/n^2)$ factors respectively, then we can solve the k-Clique problem with only polynomial overhead in running time. This implies the following lower bounds:
  1. Unless W[1] = FPT, there is no FPT-AS for Max k-VC and Min k-VC with running time $\exp(f(k)\cdot o(\log(1/\varepsilon)))\cdot \mathsf{poly}(n)$ for any function f (because this would give an FPT time algorithm for k-Clique when plugging in $\varepsilon = 1/n^2$).
  2. Unless k-Clique can be solved in $g(k)\cdot n^{o(k)}$ time for some function g, there is no FPT-AS for Max k-VC and Min k-VC with running time $O(1/\varepsilon)^{o(k)}\cdot\mathsf{poly}(n)$.
- For Max k-VC, the reduction that proves $(1+\delta)$-factor NP-hardness of approximation [49] is in fact a linear size reduction from the gap version of 3SAT. As a result, assuming the Gap Exponential Time Hypothesis (Gap-ETH)[10], there is no FPT-AS that runs in time $f(1/\varepsilon)^{o(k)}\cdot\mathsf{poly}(n)$ for any function f. Under the weaker ETH, a lower bound of the form $f(1/\varepsilon)^{o(k/\mathsf{poly}\log k)}\cdot\mathsf{poly}(n)$ for any f can be achieved via nearly linear size PCP [18]. (Note that we do not know any lower bound of this form for Min k-VC; in particular, it is not known whether Min k-VC is NP-hard to approximate even for a factor of 1.0001.)

An interesting remaining open question is to close the gap between the (polynomial time) approximation algorithms and hardness of approximation for Max k-VC. On the algorithmic front, we note that Austrin et al. [6] further exploited the techniques developed by Raghavendra and Tan [52] to achieve several improvements. Most importantly, they show that, for Max 2SAT with cardinality constraint, if the cardinality constraint is $x_1+\cdots+x_n = n/2$ (i.e. $k = n/2$), then an 0.94-approximation can be achieved in polynomial time. (In particular, the ratio here is the same as the ratio of the Lewin-Livnat-Zwick algorithm for Max 2SAT without cardinality constraint [42]; see also [5, 53]. Note that this ratio is still different from the hardness from [7].) This specific case is often referred to as *Max Bisection 2SAT*. Unfortunately, the algorithm does not naturally[11] extend to the case where $k \neq n/2$ and hence it is unclear how to employ this algorithm for Max k-VC.

On the hardness of approximation front, we remark that the hardness that follows from [7] holds even for the *perfect completeness* case. That is, even when there is a vertex cover of size k, it is still hard to find k vertices that cover 0.944 fraction of the edges. (See Appendix A of the full version [44].) Interestingly, there is an evidence that this perfect completeness case is easier: Feige and Langberg [23] shows that their algorithm achieves 0.8-approximation in this case, which is better than $(0.75+\delta)$-approximation that their algorithm yields in the

[10] Gap-ETH states that there is no $2^{o(n)}$-time algorithm that can distinguish between a fully satisfiable 3CNF formula and one which is not even 0.999-satisfiable [19, 45].

[11] In particular, the rounding algorithm involves scaling the bias of the variables (see Section 6 of [6]). For *Max Bisection 2SAT*, the sum of the bias is zero and hence scaling retains the sum. However, when the sum is non-zero, scaling changes the sum and hence the rounding algorithm produces a subset of size not equal to k.

general case. In fact, we can even get 0.94-approximation in this case as follows. First, we follow the kernelization for Vertex Cover [16] based on the Nemhauser-Trotter theorem [48]: on input graph (G, k), this gives a partition $V_0, V_{1/2}, V_1$ such that there exists a vertex cover S of size k such that $V_1 \subseteq S \subseteq V_{1/2} \cup V_1$. Moreover, the Nemhauser-Trotter theorem also ensures that $|V_{1/2}| = 2 \cdot (k - |V_1|)$. This means that we can restrict ourselves to the graph induced by $V_{1/2}$ and applies the aforementioned Max Bisection 2SAT from [6]. This indeed gives us a 0.94-approximation as desired. These suggest that it might be that the perfect completeness case is easier to approximate; thus, it would be interesting to see whether there is any way to construct harder instances with imperfect completeness.

References

1 Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christopher T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In *ALENEX*, pages 62–69, 2004.

2 Alexander A. Ageev and Maxim Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *J. Comb. Optim.*, 8(3):307–328, 2004.

3 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

4 Nicola Apollonio and Bruno Simeone. The maximum vertex coverage problem on bipartite graphs. *Discrete Applied Mathematics*, 165:37–48, 2014.

5 Per Austrin. Balanced Max 2-Sat might not be the hardest. In *STOC*, pages 189–197, 2007.

6 Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better balance by being biased: A 0.8776-approximation for max bisection. In *SODA*, pages 277–294, 2013.

7 Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory of Computing*, 7(1):27–43, 2011.

8 Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In *FOCS*, pages 453–462, 2009.

9 R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109 of *North-Holland Mathematics Studies*, pages 27 – 45. North-Holland, 1985.

10 Reuven Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. *J. Algorithms*, 39(2):137–144, 2001.

11 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

12 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.

13 Édouard Bonnet, Bruno Escoffier, Vangelis Th. Paschos, and Georgios Stamoulis. Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs. *Discrete Optimization*, 27:26–56, 2018.

14 Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993.

15 Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice classes of parameterized tractability. *Ann. Pure Appl. Logic*, 84(1):119–138, 1997.

16 Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

17 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010.

18 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

19 Irit Dinur. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:128, 2016.

20 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? *ECCC*, 23:198, 2016.

21 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. On non-optimally expanding sets in grassmann graphs. *ECCC*, 24:94, 2017.

22 Irit Dinur and Shmuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

23 Uriel Feige and Michael Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 41(2):174–211, 2001.

24 Michael R. Fellows, Ariel Kulik, Frances A. Rosamond, and Hadas Shachnai. Parameterized approximation via fidelity preserving transformations. *J. Comput. Syst. Sci.*, 93:30–40, 2018.

25 Rajiv Gandhi and Guy Kortsarz. On set expansion problems and the small set expansion conjecture. *Discrete Applied Mathematics*, 194:93–101, 2015.

26 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

27 Oded Goldreich. On promise problems: A survey. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, pages 254–290, 2006.

28 Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized complexity of vertex cover variants. *Theory Comput. Syst.*, 41(3):501–520, 2007.

29 Anupam Gupta, Euiwoong Lee, and Jason Li. Faster exact and approximate algorithms for k-cut. In *FOCS*, 2018. To appear.

30 Anupam Gupta, Euiwoong Lee, and Jason Li. An FPT algorithm beating 2-approximation for k-cut. In *SODA*, pages 2821–2837, 2018.

31 Eran Halperin and Uri Zwick. A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems. *Random Struct. Algorithms*, 20(3):382–402, 2002.

32 Qiaoming Han, Yinyu Ye, Hantao Zhang, and Jiawei Zhang. On approximation of max-vertex-cover. *European Journal of Operational Research*, 143(2):342–355, 2002.

33 Qiaoming Han, Yinyu Ye, and Jiawei Zhang. An improved rounding method and semidefinite programming relaxation for graph partition. *Math. Program.*, 92(3):509–535, 2002.

34 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

35 Dorit S. Hochbaum. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., Boston, MA, USA, 1997.

36 George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Trans. Algorithms*, 5(4):41:1–41:8, 2009.

37 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

38 Subhash Khot. On the power of unique 2-prover 1-round games. In *CCC*, page 25, 2002.

39 Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and grassmann graphs. In *STOC*, pages 576–589, 2017.

40 Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. *ECCC*, 25:6, 2018.

41 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

42 Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-sat and MAX DI-CUT problems. In *Integer Programming and Combinatorial Optimization,*

9th International IPCO Conference, Cambridge, MA, USA, May 27-29, 2002, Proceedings, pages 67–82, 2002.

43 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *STOC*, pages 224–237, 2017.

44 Pasin Manurangsi. A note on max k-vertex cover: Faster fpt-as, smaller approximate kernel and improved approximation. *arXiv preprint arXiv:1810.03792*, 2018.

45 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense csps. In *ICALP*, pages 78:1–78:15, 2017.

46 Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008.

47 Burkhard Monien and Ewald Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Inf.*, 22(1):115–123, 1985.

48 George L. Nemhauser and Leslie E. Trotter Jr. Properties of vertex packing and independence system polyhedra. *Math. Program.*, 6(1):48–61, 1974.

49 Erez Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.

50 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *STOC*, pages 755–764, 2010.

51 Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between expansion problems. In *CCC*, pages 64–73, 2012.

52 Prasad Raghavendra and Ning Tan. Approximating CSPs with global cardinality constraints using SDP hierarchies. In *SODA*, pages 373–387, 2012.

53 Henrik Sjögren. Rigorous analysis of approximation algorithms for MAX 2-CSP. Master's thesis, KTH Royal Institute of Technology, 2009.