Math. Program., Ser. A (2019) 177:1–19
https://doi.org/10.1007/s10107-018-1255-7

FULL LENGTH PAPER

Partitioning a graph into small pieces with applications to path transversal

Euiwoong Lee[1]

Received: 16 July 2017 / Accepted: 27 February 2018 / Published online: 9 March 2018

Abstract Given a graph $G = (V, E)$ and an integer $k \in \mathbb{N}$, we study k-VERTEX SEPARATOR (resp. k-EDGE SEPARATOR), where the goal is to remove the minimum number of vertices (resp. edges) such that each connected component in the resulting graph has less than k vertices. We also study k-PATH TRANSVERSAL, where the goal is to remove the minimum number of vertices such that there is no simple path of length k. Our main results are the following improved approximation algorithms.

- $O(\log k)$-approximation algorithm for k-VERTEX SEPARATOR that runs in time $2^{O(k)}n + n^{O(1)}$. It improves the simple k-approximation, and runs faster than the lower bound $k^{\Omega(\mathsf{OPT})}n^{\Omega(1)}$ for exact algorithms assuming the exponential time hypothesis (ETH) when $\mathsf{OPT} > k$. We also prove that there is no $n^{(1/\log\log n)^c}$-approximation algorithm that runs in time poly(n, k) assuming the ETH.
- $O(\log k)$-approximation algorithm for k-EDGE SEPARATOR that runs in time $n^{O(1)}$. It improves the best previous graph partitioning algorithm for small $k = n^{o(1)}$.
- $O(\log k)$-approximation algorithm for k-PATH TRANSVERSAL that runs in time $n^{O(1)} + 2^{O(k^3 \log k)}n^2 \log n$. Previously, the existence of an $(1-\delta)k$-approximation algorithm for fixed $\delta > 0$ (even using $n^{f(k)}$ time) was open.

Preliminary version of this paper appeared in SODA'17. Supported by the Samsung Scholarship, the Simons Award for Graduate Students in Theoretical Computer Science, and Venkat Guruswami's NSF CCF-1115525. Part of this work was done when the author was an intern at Microsoft Research.

✉ Euiwoong Lee
euiwoonl@cs.cmu.edu

1 Department of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Keywords Graph partitioning · Path transversal · Fixed parameter tractability · Subgraph deletion

Mathematics Subject Classification 68W25 (Approximation algorithms) · 90C27 (Combinatorial optimization)

1 Introduction

Graph partitioning is a general task of removing a small number of edges or vertices to make the resulting graph consist of smaller connected components. We study the following natural graph partitioning problems.

k-Vertex Separator
Input: An undirected graph $G = (V, E)$ and $k \in \mathbb{N}$.
Output: Subset $S \subseteq V$ such that in the subgraph induced on $V \setminus S$ (denoted by $G[V \setminus S]$), each connected component has strictly less than k vertices.
Goal: Minimize $|S|$.

The edge version can be defined similarly.

k-Edge Separator
Input: An undirected graph $G = (V, E)$ and $k \in \mathbb{N}$.
Output: Subset $S \subseteq E$ such that in the subgraph $(V, E \setminus S)$, each connected component has strictly less than k vertices.
Goal: Minimize $|S|$.

The best previous approximation algorithms achieve k-approximation for k-Vertex Separator [3] and $O(\sqrt{\log n \log(n/k)})$-approximation [26]. Assuming the Exponential Time Hypothesis (ETH), k-Vertex Separator cannot be solved in $k^{o(\mathsf{OPT})} n^{O(1)}$ time [12].

Graph partitioning has been one of the most actively studied combinatorial optimization problems. It is not only related to fundamental graph properties such as expansion and spectrum (see a course devoted to this connection [34]), but also often used as an important subroutine for Divide-and-Conquer approaches to numerous other optimization problems. In this paper, an improved algorithm for (a generalization of) k-Vertex Separator yields an improved algorithm for the following natural problem. Let $l(G)$ be the length of the longest path of G including both endpoints (e.g., length of a single edge is 2).

k-Path Transversal
Input: An undirected graph $G = (V, E)$ and $k \in \mathbb{N}$.
Output: Subset $S \subseteq V$ such that $l(G[V \setminus S]) < k$.
Goal: Minimize $|S|$.

k-Path Transversal is motivated by applications in transportation and wireless sensor networks, and has also been actively studied as k-Path Vertex Cover or P_k-Hitting Set in terms of their structural property, approximability, and fixed parameter tractability [6,7,9,24,35]. However, approximation algorithms that beat the trivial k-approximation were known only for $k = 3, 4$ [9,35], and it was open whether we can get $(1-\delta)k$-approximation for k-Path Transversal for a general k and a universal constant $\delta > 0$.

1.1 Our results and applications

For k-Vertex Separator, k-Edge Separator, and k-Path Transversal, we present improved approximation algorithms. Our focus is when k is much smaller than n. Our algorithms for the vertex problems (k-Vertex Separator and k-Path Transversal) run in time $2^{\mathrm{poly}(k)} n^{O(1)}$, so they can be regarded as fixed parameter tractable (FPT) approximation algorithms (see [30] for a survey). We also prove a hardness result shwoing that we cannot achieve polylog(n)-approximation in time poly(n, k) assuming the ETH. Our algorithm for k-Edge Separator strictly improves the previous best approximation algorithm [26] when $k = n^{o(1)}$.

Our result for k-Vertex Separator is based on the following algorithm for k-Vertex Separator. For fixed constants $b, c > 1$, an algorithm for k-Vertex Separator is called an (b, c)-bicriteria approximation algorithm if given an instance $G = (V, E)$ and $k \in \mathbb{N}$, it outputs $S \subseteq V$ such that (1) each connected component of $G[S \backslash V]$ has at most bk vertices and (2) $|S|$ is at most c times the optimum of k-Vertex Separator.

Theorem 1 *For any $\varepsilon \in (0, 1/4]$, there is an $\left(\frac{1}{1-2\varepsilon}, O\left(\frac{\log k}{\varepsilon}\right)\right)$-bicriteria approximation algorithm for k-*Vertex Separator *that runs in time $n^{O(1)}$.*

Setting $\varepsilon = \frac{1}{4}$ and running the algorithm yields $S \subseteq V$ with $|S| \leq O(\log k) \cdot \mathsf{OPT}$ such that each component in $G[V \backslash S]$ has at most $2k$ vertices. To contrast with bicriteria approximation, we often say that an algorithm achieves *true* approximation when each component in $G[V \backslash S]$ has at most k vertices as desired. Performing an exhaustive search in each connected component yields the following true approximation algorithm whose running time depends exponentially only on k.

Corollary 1 *There is an $O(\log k)$-approximation algorithm for k-*Vertex Separator *that runs in time $n^{O(1)} + 2^{O(k)} n$.*

This gives an FPT approximation algorithm when parameterized by k only, and its approximation ratio $O(\log k)$ improves the simple k-approximation [3], though the latter runs in poly(n, k) time. When $\mathsf{OPT} \gg k$, it runs even faster than the time lower bound $k^{\Omega(\mathsf{OPT})} n^{\Omega(1)}$ for the exact algorithm assuming the ETH [12].

The natural question is whether there is an algorithm that runs in time poly(n, k) and achieves true $O(\log k)$-approximation. The following theorem proves hardness of k-Vertex Separator based on Densest k-Subgraph. Densest k-Subgraph is a problem where given a graph $G = (V, E)$ and $k \in \mathbb{N}$, we want to find $S \subseteq V$, $|S| = k$ to maximize the number of edges induced by S. In particular, we show that a polynomial time $O(\log k)$-approximation algorithm for k-Vertex Separator will imply an $O(\log^2 n)$-approximation algorithm for Densest k-Subgraph. Using the recent hardness result of Manurangsi [29] that showed $n^{(1/\log\log n)^c}$-hardness of approximation for Densest k-Subgraph under the ETH, we can conclude that any polylog(n) approximation cannot be done in time poly(n, k).

Theorem 2 *If there is a* poly(n,k)*-time* f*-approximation algorithm for* k-VERTEX SEPARATOR*, then there is a* poly(n,k)*-time* $2f^2$*-approximation algorithm for* DENSEST k-SUBGRAPH. *In particular, assuming the Exponential Time Hypothesis, there is no algorithm for* k-VERTEX SEPARATOR *that runs in time* poly(n,k) *and achieves* $n^{(1/\log\log n)^c}$*-approximation for some absolute constant* $c > 0$.

For k-EDGE SEPARATOR, we prove that the true $O(\log k)$-approximation can be achieved in polynomial time. This shows a stark difference between the vertex version and the edge version.

Theorem 3 *There is an* $O(\log k)$*-approximation algorithm for* k-EDGE SEPARATOR *that runs in time* $n^{O(1)}$.

When $k = n^{o(1)}$, our algorithm outperforms the previous best $O(\sqrt{\log n \log(n/k)})$-approximation algorithm [26].

We show that k-PATH TRANSVERSAL admits $O(\log k)$-approximation in FPT time. This is the first approximation algorithm that strictly improves the simple k-approximation algorithm for general constant k. Note that the running time cannot be polynomial time k for any approximation algorithm since detecting a single k-path is NP-hard (it includes HAMILTONIAN PATH as a special case).

Theorem 4 *There is an* $O(\log k)$*-approximation algorithm for* k-PATH TRANSVERSAL *that runs in time* $2^{O(k^3 \log k)} n^2 \log n + n^{O(1)}$.

1.2 Related work

k-VERTEX SEPARATOR, k-EDGE SEPARATOR, and k-PATH TRANSVERSAL have been actively studied in a number of different research contexts that have been developed independently. We categorize past research into three groups.

Graph partitioning. In the graph partitioning literature, the edge versions have received more attention. One of the most well-studied formulations is called l-BALANCED PARTITIONING. Given a graph $G = (V, E)$ and $l \in \mathbb{N}$, the goal is to remove the smallest number of edges so that the resulting graph has l ($l \geq 2$) connected components with (roughly) the same number $\frac{n}{l}$ of vertices.[1] The case $l = 2$ has been studied extensively and produced elegant approximation algorithms. The best results are $O(\log n)$-true approximation (i.e., each component must have $\frac{n}{2}$ vertices) [33] and $O(\sqrt{\log n})$-bicriteria approximation (i.e., each component must have at most $\frac{2n}{3}$ vertices) [2]. The extension to $l \geq 3$ has been studied more recently. While it is NP-hard to achieve any nontrivial true approximation for general l [1], Krauthgamer et al. [26] presented an $O(\sqrt{\log n \log l})$-bicriteria approximation where the resulting graph is guaranteed to have each connected component with at most $\frac{2n}{l}$ vertices.

The true approximation for l-BALANCED PARTITIONING is ruled out by encoding INTEGER 3- PARTITION in graphs, and hard instances contain disjoint cliques of size[2]

[1] In the literature it is called k-BALANCED PARTITIONING. We use l in order to avoid confusion between l-BALANCED PARTITIONING and k-EDGE SEPARATOR ($l = \frac{n}{k}$).

[2] In this paper, unless otherwise stated, the size of a graph indicates the number of vertices.

at most $\frac{n}{l}$. Even et al. [13] defined a similar problem called ρ-SEPARATOR, which is our k-EDGE SEPARATOR with $\rho = \frac{k}{n}$.[3] They "believe that the definition of ρ-SEPARATOR captures type of partitioning that is actually required in applications", since "instead of limiting the number of resulting parts, which is not always important for divide-and-conquer applications or for parallelism, it limits only the sizes or weights of each part." They provided a bicriteria approximation algorithm that removes at most $O\left(\frac{1+\varepsilon}{\varepsilon}\log n\right)\cdot \mathsf{OPT}$ edges to make sure that each component has size $(1+\varepsilon)\rho n$ for any $\varepsilon > 0$, which is improved to $O\left(\frac{1+\varepsilon}{\varepsilon}\sqrt{\log(1/\varepsilon\rho)\log n}\right)\cdot \mathsf{OPT}$ by Krauthgamer et al. [26]. In our notation, it is $O\left(\frac{1+\varepsilon}{\varepsilon}\sqrt{\log(n/\varepsilon k)\log n}\right)\cdot \mathsf{OPT}$. Both [13] and [26] only presented bicriteria approximation algorithms, but the (folklore) trick presented in Sect. 4 shows that after setting $\varepsilon \le 0.5$, their algorithm can be made to true approximation algorithms by adding $O(\log k)$ to the approximation ratios. When $k = n^{o(1)}$, our approximation ratio $O(\log k) = o(\log n)$ improves $O(\sqrt{\log(n/k)\log n} + \log k) = O(\log n)$ of [26].

Some of the ideas can be used for the analogous vertex versions, but they have not received the same amount of attention. Often additional algorithmic ideas were required to achieve the same guarantee [15], or matching the same guarantee is proved to be NP-hard under some complexity assumptions [27].

Subgraph deletion. Here we focus on the vertex versions. Note that k-VERTEX SEPARATOR and k-PATH TRANSVERSAL are equivalent when $k = 2, 3, 4$ (every connected graph with at most 3 vertices has a Hamiltonian path), and these cases have been actively studied. 2-VERTEX SEPARATOR is the famous VERTEX COVER problem. When $k = 3$, Papadimitriou and Yannakakis [32,36] defined the *dissociation number* to be n minus the optimum of 3-VERTEX SEPARATOR in the context of certain constrained spanning tree problems, which have been studied independently from the graph partitioning literature (see [31] for a survey).

A simple k-approximation for k-VERTEX SEPARATOR can be achieved by viewing them as a special case of k-HYPERGRAPH VERTEX COVER (k-HVC). Given a graph $G = (V, E_G)$, we construct a hypergraph $H = (V, E_H)$ where E_H contains every set of k vertices $\{v_1, \ldots, v_k\} \subseteq V$ that induces a connected graph. This reduction is complete and sound because a subset $S \subseteq V$ intersects every hyperedge in E_H if and only if $G[V\backslash S]$ has no connected component of size at least k. Since k-HYPERGRAPH VERTEX COVER admits a simple k-approximation (e.g., take any hyperedge e not intersecting S and let $S \leftarrow S \cup e$), we get a k-true approximation for k-VERTEX SEPARATOR. This was observed in the work of Ben-Ameur et al. [3]. A k-approximation algorithm for k-PATH TRANSVERSAL can be obtained similarly.

Approximating k-HVC better than the trivial factor k (resp. $k-1$) will refute the Unique Games Conjecture (resp. $\mathbf{P} \neq \mathbf{NP}$) [11,25], so we cannot hope to be able to get a significantly better algorithm for k-HVC. An interesting line of research has tried to find a better approximation algorithm when the hypergraph H is promised to have additional structure. When H is k-uniform and k-partite, Lovász [28] gave a

[3] Except the minor difference that we want strictly less than k vertices in each component.

$\frac{k}{2}$-approximation algorithm that is shown to be tight under the Unique Games Conjecture [22] (the same work also showed almost tight $\frac{k}{2} - 1 + \frac{1}{2k}$ NP-hardness).

Given two graphs G, H where H is the *pattern graph* with k vertices, Guruswami and Lee [21] studied the problem of removing the minimum number of vertices from G such that the resulting graph has no copy of H as a subgraph. They showed that if H is 2-vertex connected, this problem is as hard to approximate as the general k-HVC. They also showed an $O(\log k)$-approximation algorithm for k-Star (a tree with $k-1$ leaves). Our result for k-PATH TRANSVERSAL gives the second example of connected H that admits $O(\log k)$-approximation for general k. The previous best approximation algorithms for k-Path guarantee a 2-approximation for $k = 3$ [35] and a 3-approximation for $k = 4$ [9].

k-VERTEX SEPARATOR can be regarded as a special case of a more general class of problems where we are given a graph G and a set of pattern graphs $\mathcal{H}$ with k vertices and asked to remove the minimum number of vertices to ensure G does not have any graph in $\mathcal{H}$ as a subgraph (in this case $\mathcal{H}$ is the set of all connected graphs with k vertices).

Generally, for a fixed graph H or a family of graphs $\mathcal{H}$, the problems of deleting the minimum number of vertices or edges from G so that G does not have any H ($\in \mathcal{H}$) as a subgraph [21]/induced subgraph [5]/minor [16]/immersion [19] have been studied actively in terms of their approximability and parameterized complexity.

Fixed parameter tractability. Given a graph G and an integer k, the optimum of k-VERTEX SEPARATOR has been known as *k-Component Order Connectivity* in mathematics. We refer the reader to the survey by Gross et al. [20] for more background.

Let OPT be the optimal value. For small values of k and OPT, the complexity of exact algorithms has been studied in terms of their fixed parameter tractability. While the trivial algorithm takes $n^{O(\mathsf{OPT})}$ time to find the exact solution for k-VERTEX SEPARATOR, Drange et al. [12] presented an exact algorithm that runs in time $k^{O(\mathsf{OPT})}n$, so the problem is in FPT when parameterized by both k and OPT. They complemented their result by showing that the problem is $\mathbf{W}[1]$-hard when parameterized by only one of them. They also showed that any exact algorithm that runs in time $k^{o(\mathsf{OPT})}n^{O(1)}$ will refute the Exponential Time Hypothesis.

Unlike k-VERTEX SEPARATOR, it is not trivial to check whether a given $S \subseteq V$ is a valid k-PATH TRANSVERSAL. Finding a path of length k has played a central role in development of FPT algorithms—it is NP-hard to do for general k, but there are various randomized and deterministic algorithms that run in time $2^{O(k)}n^{O(1)}$ (see [17] for a survey). The parameterized complexity of k-PATH TRANSVERSAL is not as well understood as k-VERTEX SEPARATOR. VERTEX COVER ($k = 2$) admits an exact FPT algorithm parameterized by OPT, which was extended to $k = 3$ [24].

Given an instance of a problem with a parameter κ, an approximation algorithm is said to be an FPT c-approximation algorithm if it runs in time $f(\kappa) \cdot n^{O(1)}$ for some function f and achieves c-approximation. See the survey of Marx [30] and the recent work of Chitnis et al. [10]. For k-VERTEX SEPARATOR, the simple k-approximation runs in polynomial time regardless of OPT and k, but any exact algorithm requires both OPT and k to be parameterized. Our algorithms for k-VERTEX SEPARATOR and k-PATH TRANSVERSAL are FPT approximation algorithms in this sense.

2 Techniques

Our algorithms for k-VERTEX SEPARATOR and k-EDGE SEPARATOR combine a previously considered linear programming (LP) relaxation and a rounding algorithm developed in different contexts. Our algorithm for k-PATH TRANSVERSAL is achieved by establishing a close connection to a generalization of k-VERTEX SEPARATOR called k-SUBSET VERTEX SEPARATOR formally defined in Sect. 3.

2.1 *k*-VERTEX SEPARATOR

Our algorithms for k-VERTEX SEPARATOR and k-EDGE SEPARATOR consist of the following three steps. We give a simple overview of our techniques for k-VERTEX SEPARATOR.

1. *Spreading metrics. Spreading metrics* were introduced in Even et al. [14] and subsequently used for ρ-separator [13].[4] They assign lengths to vertices such that any subset S of vertices with $|S| \geq k$ that induce a single connected component is *spread apart.*
   Given lengths x_v to each vertex $v \in V$, define $d_{u,v}$ to be the smallest sum of the lengths of vertices along any path from u to v, including the lengths of both u and v (so that $d_{u,u} = x_u$). Given a feasible solution $S \subseteq V$ for k-VERTEX SEPARATOR, let $x_v = 1$ if $v \in S$, and $x_v = 0$ if $v \notin S$. It is easy to see that two vertices u and v lie on the same component of $G[V \backslash S]$ if and only if $d_{u,v} = 0$. Otherwise, $d_{u,v} \geq 1$. Therefore, for every vertex v, the number of vertices that have distance strictly less than 1 from v must be strictly less than k.
   Spreading metrics are a continuous relaxation of the above integer program. We relax each distance x_v to have value in $[0, 1]$, and let $d_{u,v}$ still be the length of the shortest path from u to v. Let $f_{u,v} = \max(1 - d_{u,v}, 0)$. In the integral solution, it indicates whether $d_{u,v} < 1$ or not. The constraint $\sum_u f_{v,u} \leq k - 1$ for all $v \in V$ is a relaxation of the requirement that the number of vertices that have distance strictly less than 1 from v must be strictly less than k.
   Even though this relaxation does not exactly capture the integer problem, one crucial property of this relaxation is that for every $v \in V$ and $\varepsilon \in (0, \frac{1}{4}]$, the number of vertices that have distance at most ε from v can be at most $\frac{k}{1-2\varepsilon}$. This can proved via a simple averaging argument.
2. *Low-diameter decomposition.* Before we introduce our rounding algorithm, we briefly discuss why the previous algorithms based on the same (or stronger) relaxation have the approximation ratio depending on n.
   The current best algorithm by Krauthgamer et al. [26] further strengthened the above spreading metrics by requiring that they also form an ℓ_2^2 metric, and trans-

[4] The conference version of [14] precedes that of [13].

formed them to an ℓ_2 metric. This black-box transformation of an n-points ℓ_2^2 metric incurs distortion of $\Omega(\sqrt{\log n})$, so the approximation ratio must depend on n.
The older work of Even et al. [13] used the rounding algorithm of Garg et al. [18] that iterative takes a ball of small radius from the graph. More specifically, they defined $\mathsf{vol}(v, r)$ to be the total sum of lengths in the ball of radius r centered at v, and grow r until the boundary-volume ratio becomes $O\left(\log\left(\frac{\mathsf{vol}(v,\frac{1}{2})}{\mathsf{vol}(v,0)}\right)\right)$. To make $\mathsf{vol}(v, 0)$ nonzero, a *seed value* of $\varepsilon \cdot \mathsf{OPT}$ must be added to the the definition of $\mathsf{vol}(v, r)$. But when $k = O(1)$ so that the number of balls we need to remove from the graph is $\Omega(n)$, this incurs extra cost of $\Omega(\varepsilon n \mathsf{OPT})$, forcing ε to depend on n.
We apply another standard technique for the *low-diameter decomposition* to our spreading metrics. In particular, our algorithm is similar to that of Calinescu et al. [8], preceded by a simple rounding algorithm that removes every vertex with large x_v. One simple but crucial observation is that the performance of this algorithm only depends on the size of the ball around each vertex, which is exactly what spreading metrics is designed for! Since the size of each ball of radius $\frac{1}{4}$ is at most $O(k)$, we can guarantee that we can delete at most $O(\log k) \cdot \mathsf{OPT}$ vertices so that each connected component has at most $O(k)$ vertices.
When $k = O(1)$, to the best of our knowledge, this is a rare example where the number of partitions (i.e., the number of balls taken) is $\Omega(n)$ but the approximation ratio is much smaller than that. The original rounding algorithm of Calinescu et al. [8] is applied to 0-*Extension* with k terminals to achieve $O(\log k)$-approximation, where only k balls are needed to be taken. The famous $O(\log k)$-approximation for Multicut with k source-sink pairs [18] also required only k partitions.

3. *Cleanup.* After running the bicriteria approximation algorithm to make sure that each connected component has size at most $O(k)$, for k-Vertex Separator, we perform an exhaustive search in each component to get a true approximation overall. This incurs the extra running time of $2^{O(k)}n$, but our hardness result implies that we cannot get polylog(n, k)-approximation in time poly(n, k).

For k-Edge Separator, essentially the same bicriteria approximation algorithm works. After that, for each component, we use (a variant of) Racke's $O(\log n)$-true approximation algorithm for Minimum Bisection [33] to each component to make sure that each component has strictly less than k vertices. The existence of a *true approximation* for Minimum Bisection is a key difference between the vertex version and the edge version. Even an $O(\sqrt{\log n})$-bicriteria approximation is known for the vertex version of Minimum Bisection [15], but our hardness result for the vertex version suggests that this algorithm is not likely to be applicable. While Minimum Bisection asks to partition the graph into two pieces and k-Edge Separator may need to partition it into many pieces, we prove that as long as each connected component has size at most $\frac{3k}{2}$, a simple trick makes the two problems equivalent.

2.2 k-Path Transversal

Note that $S \subseteq V$ is a valid solution for k-Vertex Separator if it is for k-Path Transversal, simply because a graph cannot have a k-path if every connected component has at most $k - 1$ vertices. Therefore, $\mathsf{OPT_S} \geq \mathsf{OPT_P}$ when $\mathsf{OPT_S}$ and $\mathsf{OPT_P}$ denote the optimal value of k-Vertex Separator and k-Path Transversal respectively.

Let $S^* \subseteq V$ be the optimal k-Path Transversal. Ideally, if S^* is also a valid k-Vertex Separator (which implies that it is also optimal for k-Vertex Separator), then we can use our $O(\log k)$-approximation algorithm for k-Vertex Separator to find a $S \subseteq V$ with $|S| \leq (\log k)|S^*|$, which is a valid k-Vertex Separator and k-Path Transversal.

Of course, this ideal situation is not always true because the subgraph $G[V \setminus S^*]$ is only guaranteed not to have a k-path and each connected component can be arbitrarily large. For example, a star graph (a tree where all but one vertex are leaves) does not even have a 4-path but can have arbitrarily large number of vertices. But it is also *fragile* in the sense that deleting one vertex results in many connected components with one vertex. Therefore, we can hope that if we delete S^* and some more vertices, the graph is guaranteed to break into many small components. This implies $\mathsf{OPT_P} \leq O(\mathsf{OPT_S})$ so that we can use the previous argument!

Our algorithm for k-Vertex Separator formalizes this intuition in an indirect way. The first step of the algorithm is to compute a k-approximation solution $R \subseteq V$. Call them *red* vertices. Our key lemma proves that if we delete a set S' of at most $\mathsf{OPT_P}$ more vertices on top of S^*, the graph $G[V \setminus (S^* \cup S')]$ has at most k^3 red vertices in each connected component.

At this point, we introduce a generalization of k-Vertex Separator called k-Subset Vertex Separator where given a graph G and a (arbitrary) subset R, the goal is to delete the minimum number of vertices so that each connected component has less than k vertices from R. While it seems a nontrivial generalization of k-Vertex Separator, we prove that the same technique gives an FPT $O(\log k)$-approximation algorithm for k-Subset Vertex Separator.

Going back to k-Path Transversal, the key lemma ensures that the optimal value of k^3-Subset Vertex Separator is at most $2\mathsf{OPT_P}$ (as certified by $S^* \cup S'$). Then we run our algorithm for k^3-Subset Vertex Separator to delete at most $O(\log k^3) \cdot 2\mathsf{OPT_P} = O(\log k)\mathsf{OPT_P}$ vertices so that each connected component has at most k^3 red vertices. Since R is a valid solution for k-Vertex Separator, it means that the optimal value is at most k^3 in each connected component. We use a simple FPT algorithm to optimally find the k-Path Transversal in each connected component.

Organization. We prove our result for k-Subset Vertex Separator in Sect. 3 that implies our results for k-Vertex Separator. Sections 4 and 5 present our results for k-Edge Separator and k-Path Transversal respectively. Section 6 contains our hardness result for k-Vertex Separator.

3 k-Subset Vertex Separator

In this section, we present an algorithm for the vertex-weighted version of k-Subset Vertex Separator, a generalization of k-Vertex Separator. Introducing vertex weights will be helpful when we use this algorithm for k-Edge Separator. We formally define k-Subset Vertex Separator.

k-Subset Vertex Separator
Input: An undirected graph $G = (V, E)$ with vertex weights $\{w_v\}_{v \in V}$, a subset $R \subseteq V$, and $k \in \mathbb{N}$.
Output: Subset $S \subseteq V$ such that in $G[V \setminus S]$, each connected component has strictly less than k vertices from R.
Goal: Minimize $w(S) := \sum_{v \in S} w_v$.

In this section, we prove the following theorem that implies Theorem 1 and Corollary 1 for k-Vertex Separator as corollaries (by taking $R = V$).

Theorem 5 *For any $\varepsilon \in (0, 1/4]$, there is a polynomial time $(\frac{1}{1-2\varepsilon}, O(\frac{\log k}{\varepsilon}))$-bicriteria approximation algorithm for k-Subset Vertex Separator.*

3.1 Spreading metrics

Our relaxation is close to *spreading metrics* used for ρ-separator [13]. While their relaxation involves an exponential number of constraints and is solved by the ellipsoid algorithm, we present a simpler relaxation where the total number of variables and constraints is polynomial. Given a graph $G = (V, E)$, $k \in \mathbb{N}$, and a subset $R \subseteq V$ of *red* vertices, our relaxation has the following variables.

- x_v for $v \in V$: It indicates whether v is removed or not.
- $d_{u,v}$ for $(u, v) \in V \times V$: Given $\{x_v\}_{v \in V}$ as lengths on vertices, $d_{u,v}$ is supposed to be the minimum distance between u and v. Let $\mathcal{P}_{u,v}$ be the set of simple paths from u to v, and given $P = (u_0 := u, u_1, \ldots, u_p := v) \in \mathcal{P}_{u,v}$, let $d(P) = x_{u_0} + \cdots + x_{u_p}$. Formally, we want

$$d_{u,v} = \min_{P \in \mathcal{P}_{u,v}} d(P).$$

  Note that $d_{u,v} = d_{v,u}$ and $d_{u,u} = x_u$.
- $f_{u,v}$ for all $(u, v) \in V \times V$: It indicates whether u and v belong to the same connected component or not.

Our LP is written as follows.

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} w_v x_v & & \\ \text{subject to} \quad & d_{u,v} \le \min_{P \in \mathcal{P}_{u,v}} d(P) & \forall (u, v) \in V \times V \qquad & (1) \\ & f_{u,v} \ge 1 - d_{u,v} & \forall (u, v) \in V \times V \qquad & \end{aligned}$$

$$
\begin{aligned}
& f_{u,v} \geq 0 && \forall (u,v) \in V \times V \\
& \sum_{u \in R} f_{v,u} \leq k-1 && \forall v \in V \\
& x_v \geq 0 && \forall v \in V
\end{aligned} \tag{2}
$$

(1) can be formally written as

$$
\begin{aligned}
& d_{u,u} = x_u && \forall u \in V \\
& d_{u,w} \leq d_{u,v} + x_w && \forall (u,v) \in V \times V, (v,w) \in E
\end{aligned}
$$

Therefore, the size of our LP is polynomial in n. It is easy to verify that our LP is a relaxation—given a subset $S \subseteq V$ such that each connected component of $G[V \backslash S]$ has less than k red vertices, the following is a feasible solution with $\sum_v x_v = w(S)$.

- $x_v = 1$ if $v \in S$. $x_v = 0$ if $v \notin S$.
- $d_{u,v} = \min_{P \in \mathcal{P}_{u,v}} d(P)$.
- $f_{u,v} = 1$ if u and v are in the same component of $G[V \backslash S]$. Otherwise $f_{u,v} = 0$.

Fix an optimal solution $\{x_v\}_v$, $\{d_{u,v}, f_{u,v}\}_{u,v}$ for the above LP. It only ensures that $d_{u,v} \leq \min_{P \in \mathcal{P}_{u,v}} d(P)$, so a priori $d_{u,v}$ can be strictly less than $\min_{P \in \mathcal{P}_{u,v}} d(P)$. However, in that case increasing $d_{u,v}$ still maintains feasibility, since larger $d_{u,v}$ provides a looser lower bound of $f_{u,v}$ and lower $f_{u,v}$ helps to satisfy (2). For the subsequent sections, we assume that $d_{u,v} = \min_{P \in \mathcal{P}_{u,v}} d(P)$, and $f_{u,v} = \max(1 - d_{u,v}, 0)$ for all u, v.

3.2 Low-diameter decomposition

Given the above spreading metrics, we show how to decompose a graph such that each connected component has a small number of vertices, proving Theorem 5. Our algorithm is based on that of Calinescu et al. [8]. One major difference is to bound the size of each *ball* by $O(k)$ in the analysis, and simple algorithmic steps to ensure this fact.

Fix $\varepsilon \in (0, \frac{1}{4}]$. Given an optimal solution $\{x_v\}_{v \in V}$, the first step of the rounding algorithm is to remove every vertex $v \in V$ with $x_v \geq \varepsilon$. This simple step is crucial in bounding the size of the ball around each vertex. Since the total weight of the vertices with $x_v \geq \varepsilon$ is at most $\mathsf{FRAC}/\varepsilon$ where FRAC denotes the value of the optimal LP solution, the total weight of the removed vertices is at most OPT/ε.

Let $V' := V \backslash \{v : x_v \geq \varepsilon\}$, and $G' = (V', E')$ be the subgraph of G induced by V'. Let $R' = V' \cap R$. Let $d'_{u,v}$ be the minimum distance between u and v in G', and let $f'_{u,v} := \max(1 - d'_{u,v}, 0)$. Since removing vertices only increases distances, $d'_{u,v} \geq d_{u,v}$ and $f'_{u,v} \leq f_{u,v}$ for all $(u, v) \in V' \times V'$.

Our low-diameter decomposition removes vertices of total weight at most $O(\frac{\log k}{\varepsilon}) \cdot \sum_{v \in V'} x_v w_v$ vertices so that each resulting connected component has at most $\frac{k}{1-2\varepsilon}$ red vertices. It proceeds as follows.

- Pick $t \in [\varepsilon/2, \varepsilon]$ uniformly at random.
- Choose a permutation $\pi : R' \mapsto R'$ uniformly at random.
- Consider the red vertices one by one, in the order given by π. Let w be the considered vertex (we consider every vertex whether it was previously disconnected, removed or not).
  - For each vertex $v \in V'$ with $d'_{w,v} - x_v \le t \le d'_{w,v}$, we remove v when it was neither removed nor *disconnected* previously.
  - The vertices in $\{v : d'_{w,v} < t\}$ are now disconnected from the rest of the graph. Say these vertices are *disconnected.*

For each vertex $w \in V'$, let $B(w) := \{v \in R' : d'_{w,v} \le 2\varepsilon\}$. A simple averaging argument bounds $|B(w)|$.

Lemma 1 *For each vertex $w \in V'$, $|B(w)| \le \frac{k}{1-2\varepsilon}$.*

Proof Assume towards contradiction that $|B(w)| > \frac{k}{1-2\varepsilon}$. For all $u \in B(w)$,

$$f_{w,u} \ge f'_{w,u} \ge 1 - d'_{w,u} \ge 1 - 2\varepsilon.$$

Furthermore, even for $u \notin B(w)$, our LP ensures that $f_{w,u} \ge 0$. Therefore,

$$\sum_{u \in R} f_{w,u} \ge \sum_{u \in B(w)} f_{w,u} \ge (1 - 2\varepsilon)|B(w)| > k,$$

contradicting (2) of our LP. □

Note that at the end of the algorithm, every red vertex is removed or disconnected, since every $w \in V'$ becomes removed or disconnected after being considered. Moreover, each connected component is a subset of $\{v : d'_{w,v} < t\}$ for some $w \in V'$ and $t \le \varepsilon$, which is a subset of $B(w)$. Therefore, each connected component has at most $\frac{k}{1-2\varepsilon}$ red vertices. We finally analyze the probability that a vertex $v \in V'$ is removed.

Lemma 2 *The probability that $v \in V'$ is removed is at most $O(\frac{\log k}{\varepsilon}) \cdot x_v$.*

Proof Fix a vertex $v \in V'$. When $w \in R'$ is considered, v can be possibly removed only if

$$\begin{aligned} & d'_{v,w} - x_v \le \varepsilon \\ \Rightarrow\ & d'_{v,w} \le 2\varepsilon \qquad \text{(since } x_v \le \varepsilon) \\ \Rightarrow\ & w \in B(v). \end{aligned}$$

Let $W = \{w_1, \ldots, w_p\}$ be such vertices such that $d'_{v,w_1} \le \cdots \le d'_{v,w_p} \le 2\varepsilon$. By Lemma 1, $p \le \frac{k}{1-2\varepsilon}$.

Fix i and consider the event that v is removed when w_i is considered. This happens only if $d'_{v,w_i} - x_v \le t \le d'_{v,w_i}$. For fixed such t, a crucial observation is that if w_j with $j < i$ is considered before w_i, since $d'_{v,w_j} - x_v \le t$, v will be either removed or disconnected when w_j is considered. In particular, v will not be removed by w_i. Given these observations, the probability that v is removed is bounded by

$$\begin{aligned}
&\Pr[v \text{ is removed}]\\
&\quad= \sum_{i=1}^{p} \Pr[v \text{ is removed when } w_i \text{ is considered}]\\
&\quad\le \sum_{i=1}^{p} \Pr[t \in [d'_{v,w_i} - x_v, d'_{v,w_i}] \text{ and } w_i \text{ comes before } w_1, \ldots, w_{i-1} \text{ in } \pi]\\
&\quad\le \sum_{i=1}^{p} \frac{2x_v}{\varepsilon i} = x_v \cdot O\left(\frac{\log p}{\varepsilon}\right) = x_v \cdot O\left(\frac{\log k}{\varepsilon}\right),
\end{aligned}$$

where the last inequality follows from the fact that $\Pr[t \in [d'_{v,w_i} - x_v, d'_{v,w_i}]] = 2x_v/\varepsilon$ (t is sampled from $[\varepsilon/2, \varepsilon]$), $\Pr[w_i \text{ comes before } w_1, \ldots, w_{i-1} \text{ in } \pi] = 1/i$, and these two events are independent. ◻

Therefore, the low-diameter decomposition removes at most $O(\frac{\log k}{\varepsilon}) \cdot \sum_v x_v \le O(\frac{\log k}{\varepsilon}) \cdot \mathsf{OPT}$ vertices so that each resulting connected component has at most $\frac{k}{1-2\varepsilon}$ red vertices. This gives a bicriteria approximation algorithm that runs in time poly(n, k). We finally show how to make the algorithm deterministic, finishing the proof of Theorem 5.

Derandomization. Note that our algorithm samples the following two random variables: $t \in [\varepsilon/2, \varepsilon]$ and $\pi : R' \mapsto R'$. Sampling t can be easily derandomized, because the only way t is used in the algorithm is to check whether $t \in [d'_{w,v} - x_v, d'_{w,v}]$ for some $(w, v) \in V^2$. There are at most n^2 such intervals and $2n^2$ endpoints $\cup_{u,v}\{d'_{w,v} - x_v, d'_{w,v}\}$. Sort these $2n^2$ numbers along with $\varepsilon/2, \varepsilon$ in increasing order. As long as t belongs to an interval formed by two consecutive numbers, the execution of our algorithm does not change, so we can just try every interval formed by these consecutive numbers and output the best outcome.

Once t is fixed, we use the standard method of conditional expectation to derandomize the choice of π. For each $i = 1, \ldots, n$, we try each remaining vertex as a possible candidate of the ith vertex in π, compute the conditional expected value of the algorithm given the first i vertices, and fix the one with the minimum value as the ith vertex and proceed. Given t and the first i vertices, the probability that a given vertex v is deleted can be efficiently computed following the proof of Lemma 2.

4 Algorithm for k-Edge Separator

We present an $O(\log k)$-true approximation algorithm for k-Edge Separator, proving Theorem 3. We fist claim that we can obtain an bicriteria approximation algorithm for k-Edge Separator from the bicriteria approximation algorithm for k-Subset Vertex Separator.

Lemma 3 *For any $\varepsilon \in (0, 1/4]$, there is an $(\frac{1}{1-2\varepsilon}, O(\frac{\log k}{\varepsilon}))$-bicriteria approximation algorithm for k-*Edge Separator *that runs in time $n^{O(1)}$.*

Proof Given an instance $G = (V, E)$ and $k \in \mathbb{N}$ for k-Edge Separator, we construct an instance $G' = (V', E'), \{w_v\}_{v \in V'}, R \subseteq V'$, and $k \in \mathbb{N}$ for the vertex-weighted k-Subset Vertex Separator by *subdividing each edge* as follows.

- $V' = V \cup E$.
- $E' = \{(u, e) : u \in V, e \in E, u \in e\}$.
- $w_v = \infty$ for all $v \in V$ (alternatively, set $x_v = 0$ when we solve the LP for k-Subset Vertex Separator so that v is never deleted).
- $w_e = 1$ for $e \in E$.
- $R = V$, and k stays the same.

There is one-to-one correspondence between the feasible solutions of k-Edge Separator and the feasible solutions of k-Subset Vertex Separator that only remove $e \in E$. By construction, any feasible solution of k-Subset Vertex Separator with finite value removes only $e \in E$. Therefore, Theorem 5 for k-Subset Vertex Separator implies the lemma. □

To get true approximation, we use the algorithm for b-Balanced Cut. For an undirected graph $G = (V, E)$ with n vertices and a real $b \in (0, 1/2]$, the b-Balanced Cut problem asks to find a subset $S \subseteq V$ with $bn \le |S| \le (1 - b)n$ such that the number of edges that have exactly one endpoint in S is minimized. Racke [33] gave an $O(\log n)$-true approximation algorithm for b-Balanced Cut.[5]

We set $\varepsilon = \frac{1}{6}$ such that each connected component after the low-diameter decomposition, each connected component has less than $\frac{3k}{2}$ vertices. Fix a component of size k'. If $k' < k$, we are done. Otherwise, we use the $O(\log k') = O(\log k)$-approximation algorithm for b-Balanced Cut within the component. Usually k-Edge Separator (requires many connected components) and b-Balanced Cut (requires 2 connected components) behave very differently, but given $k' < \frac{3k}{2}$, we show that they are equivalent.

Lemma 4 *In a graph $G = (V, E)$ with at most $k' \in (k, \frac{3}{2}k)$ vertices, the optimum solution of k-*Edge Separator *and b-*Balanced Cut *with $b = \frac{k'-k+1}{k'}$ are the same.*

Proof Any cut $(S, V \backslash S)$ feasible for b-Balanced Cut ensures that $\max(|S|, |V \backslash S|)$ is at most $(1 - b)k' = k - 1$, so it is feasible for k-Edge Separator.

For the other direction, given a feasible solution of k-Edge Separator where V is partitioned into $S_1, \ldots, S_l$ (assume $k \ge |S_1| \ge \cdots \ge |S_l|$), if $l = 2$, (S_1, S_2) is a feasible solution for b-Balanced Cut and we are done. If $l \ge 3$, merge S_{l-1}, S_l

[5] His algorithm is originally stated for Minimum Bisection, the special case with $b = \frac{1}{2}$. For any $c \in [0, 1 - 2b]$, adding a disjoint clique with cn vertices and infinite-weight edges (his algorithm works in weighted version), forces the Minimum Bisection algorithm to output a cut in the original graph where the smaller side contains exactly $\frac{(1-c)n}{2} \in [bn, \frac{n}{2}]$ vertices. Trying every value of $c \in [0, 1 - 2b]$ that makes cn an integer and taking the best cut gives the desired $O(\log n)$-true approximation for b-Balanced Cut. The author thanks to Anupam Gupta for this idea.

into one set (one S_i may contain multiple connected components). This reduces l by 1, and since $|S_{l-1}| + |S_l| \leq \frac{2}{l} \cdot k' \leq \frac{2}{3}k' < k$, maintains the invariant that $|S_i| < k$ for all i. Iterating until $l = 2$ gives a feasible solution for b-Balanced Cut with the same number of edges cut. □

Therefore, running the approximation algorithm b-Balanced Cut for each component guarantees that we remove $O(\log k) \cdot \mathsf{OPT}$ additional edges and each component has less than k vertices. This proves Theorem 3.

5 *k*-Path Transversal

Let $G = (V, E)$ and $k \in \mathbb{N}$ be an instance of k-Path Transversal, where we want to find the smallest $S \subseteq V$ such that the length of the longest path in $G[V \backslash S]$ (denoted by $l(G[V \backslash S])$) is strictly less than k. Recall that the length here denotes the number of vertices in a path. Call a path *l-path* if it has l vertices.

The first step of our algorithm is to compute an k-approximation solution $R \subseteq V$ by iteratively finding a k-path p in the graph $G[V \backslash R]$ and add p to R until there is no k-path. Since finding a k-path takes time $2^{O(k)} n \log n$ [17], this step can be implemented in time $2^{O(k)} n^2 \log n$.

Let S^* be the optimal solution of k-Path Transversal. Let $V^* := V \backslash S^*$, $R^* := R \backslash S^*$ and $G^* = G[V \backslash S^*]$. The result for k-Path Transversal requires the following lemma.

Lemma 5 *There exists $S' \subseteq V^*$ with $|S'| \leq \frac{|R^*|}{k}$ so that in the induced subgraph $G^*[V^* \backslash S']$, each connected component has at most k^3 red vertices.*

Proof We prove the lemma by the following (possibly exponential time) algorithm: For each connected component C that has more than k^3 red vertices, take an arbitrary longest path, remove all vertices in it (i.e., add them to S') and charge its cost to all red vertices in C uniformly. Since the length of any longest path should not exceed k and C has more than k^3 red vertices, each red vertex in C gets charged at most $\frac{1}{k^2}$ in each iteration.

We argue that each vertex in G^* is charged at most k times. This is based on the fact that in a connected component C, any two longest paths should intersect. Therefore, if we remove one longest path from C, whether the remaining graph is still connected or divided into several connected components, the length of the longest path in each resulting connected component should be strictly less than the length of the longest path in C. Therefore, each vertex in G^* can be charged at most k times, and the total amount of charge is $k \cdot \frac{1}{k^2} = \frac{1}{k}$. □

Consider $S^* \cup S'$. Since R is a k-approximate solution, $|S^* \cup S'| \leq \mathsf{OPT} + \frac{|R^*|}{k} \leq 2\mathsf{OPT}$, and each component of $G[S^* \cup S']$ has at most k^3 red vertices. Run the bicriteria approximation algorithm for k-Subset Vertex Separator in Theorem 5 with $k \leftarrow k^3$ and $\varepsilon \leftarrow \frac{1}{4}$. This returns a subset $S \subseteq V$ such that $|S| \leq O(\log k) \cdot \mathsf{OPT}$ and each connected component of $G_{V \backslash S}$ has at most $2k^3$ red vertices.

Now we solve the problem for each connected component C. Since every k-path has to have at least one red vertex, removing every red vertex destroys every k-path.

In particular, the optimal solution has at most $2k^3$ vertices in C. We run the following simple recursive algorithm.

- Find a k-path $P = (v_1, \ldots, v_k)$ if exists.
  - Otherwise, we found a solution—compare with the current best one and return.
- If the depth of the recursion is more than $2k^3$, return.
- For each $1 \le i \le k$,
  - Remove v_i from the graph and recurse.

Finding a path takes time $2^{O(k)} n \log n$ [17]. In each stage the algorithm makes k branches, but the depth of the recursion is at most $2k^3$ and the algorithm is guaranteed to find the optimal solution. Since the number of connected components is at most n, it runs in time $(2^{O(k)} n \log n) \cdot (k^{2k^3}) \cdot n = 2^{O(k^3 \log k)} n^2 \log n$. This proves Theorem 4.

6 Hardness of k-Vertex Separator

In this section, we prove that an f-true approximation algorithm for k-Vertex Separator that runs in time poly(n, k) will result in $2f^2$-approximation algorithm for Densest k-Subgraph, proving Theorem 2. In particular, $O(\log k)$-true approximation for k-Vertex Separator in time poly(n, k) will lead to $O(\log^2 n)$-approximation for Densest k-Subgraph.

Given an undirected graph $G = (V, E)$ and an integer k, Densest k-Subgraph asks to find $S \subseteq V$ with $|S| = k$ to maximize the number of edges of $G[S]$. The current best approximation algorithm achieves $\approx O(n^{1/4})$-approximation [4]. Using the recent hardness result of Manurangsi [29] that showed $n^{(1/\log\log n)^c}$-hardness of approximation for Densest k-Subgraph under the ETH, we can conclude that any polylog(n) approximation cannot be done in time poly(n, k) unless the ETH is false.

Our reduction is close to that of Drange et al. [12] who reduced Clique to k-Vertex Separator to prove **W**[1]-hardness. Formally, we introduce another problem called Minimum k-Edge Coverage. Given an undirected graph G and an integer k, the problem asks to find the minimum number of vertices whose induced subgraph has at least k edges. This problem can be thought as a *dual* of Densest k-Subgraph in a sense that given the same input graph, the optimum of Densest a-Subgraph is at least b if and only if the optimum of Minimum b-Edge Coverage is at most a. Hajiaghayi and Jain [23] proved the following theorem, relating their approximation ratios.

Theorem 6 [23] *If there is a polynomial time f-approximation algorithm for* Minimum k-Edge Coverage, *then there is a polynomial time $2f^2$-approximation algorithm for* Densest k-Subgraph.

We introduce a reduction from Minimum k-Edge Coverage to k-Vertex Separator. Given an instance $G = (V, E)$ and k for Minimum k-Edge Coverage, the instance of k-Vertex Separator $G' = (V', E')$ and k' is created as follows. Let $n = |V|$, $m = |E|$, and $M = n + 1$.

- $V' = V \cup \{e_i : e \in E, i \in [M]\}$. Note that $|V'| = n + Mm$.

- $E' = \binom{V}{2} \cup \{(u, e_i) : u \in V, e \in E, u \in e, i \in [M]\}$. Intuitively, the subgraph induced by $V \subseteq V'$ forms a clique, and for each $e = (u, v) \in E$ and $i \in [M]$, e_i is connected to u and v in G'.
- $k' = |V'| - Mk$.

Lemma 6 *Every instance $G' = (V', E')$ of k-*Vertex Separator *produced by the above reduction has an optimal solution $S \subseteq V'$ such that indeed $S \subseteq V$.*

Proof Take an optimal solution S to k-Vertex Separator. Suppose S contains e_i for some $e = (u, v) \in E$ and $i \in [M]$. There are three cases.

- $u, v \notin S$: Since there is an edge $(u, v) \in E'$, u and v are in the same connected component in $G'[V' \backslash S]$. Removing e_i from S and adding u to S still results in an optimal solution.
- $u \in S, v \notin S$: Removing e_i from S and adding v to S decreases the size of the connected component of v by 1, and creates a new singleton component consisting e_i. It is still an optimal solution.
- $u, v \in S$: Removing e_i from S just creates a new singleton component consisting e_i. It is a strictly better solution.

We can repeatedly apply one of these three operations until S is an optimal solution contained in V. □

When $S \subseteq V$, $G'[V' \backslash S]$ has the following connected components.

- One component $(V \backslash S) \cup \{e_i : e = (u, v) \in E, \{u, v\} \nsubseteq S, i \in [M]\}$. Call it the *giant component*.
- For each $e = (u, v) \in E$ with $u, v \in S$ and $i \in [M]$, a singleton component $\{e_i\}$. Call them *singleton components*.

Suppose that the instance G of Minimum k-Edge Coverage admits a solution $T \subseteq V$ such that the induced subgraph $G[T]$ has $l \geq k$ edges. Let $S = T$. Since $|V \backslash S| = n - |T|$ and $|\{(u, v) \in E : \{u, v\} \nsubseteq T\}| = m - l$, in $G'[V' \backslash S]$, the giant component will have cardinality

$$n - |T| + M(m - l) \leq n - |T| + M(m - k) < n + M(m - k) = |V'| - Mk = k'.$$

On the other hand, suppose that the instance G' of k'-Vertex Separator has a solution S. By Lemma 6, assume that $S \subseteq V$. Let l be the number of edges in $G[S]$. The size of the giant component is at least $n - |S| + M(m - l) > n + M(m - l - 1)$ since $M > n \geq |S|$. Since S is a feasible solution of the k'-Vertex Separator, we must have

$$\begin{aligned} &n + M(m - l - 1) < k' = n + M(m - k) \\ &\Rightarrow l \geq k. \end{aligned}$$

Therefore, S is also a solution to Minimum k-Edge Coverage. This proves that the above reduction is an approximation preserving reduction from Minimum k-Edge Coverage to k-Vertex Separator, proving Theorem 2.

Acknowledgements We thank Anupam Gupta, Guru Guruganesh, Venkatesan Guruswami, Konstantin Makarychev, and Tselil Schramm for useful discussions. We are also grateful to the anonymous reviewers for the suggestions that improved some of our algorithms and the manuscript.

References

1. Andreev, K., Racke, H.: Balanced graph partitioning. Theory Comput. Syst. **39**(6), 929–939 (2006)
2. Arora, S., Rao, S., Vazirani, U.: Expander flows, geometric embeddings and graph partitioning. J. ACM **56**(2), 5 (2009)
3. Ben-Ameur, W., Mohamed-Sidi, M.A., Neto, J.: The k-separator problem: polyhedra, complexity and approximation results. J. Comb. Optim. **29**(1), 276–307 (2015)
4. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, pp. 201–210. ACM (2010)
5. Bliznets, I., Cygan, M., Komosa, P., Pilipczuk, M.: Hardness of approximation for H-free edge modification problems. In: Jansen, K., Mathieu, C., Rolim, J.D.P., Umans, C. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016). Leibniz International Proceedings in Informatics (LIPIcs), vol. 60. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2016)
6. Brear, B., Jakovac, M., Katreni, J., Semaniin, G., Taranenko, A.: On the vertex k-path cover. Discrete Appl. Math. **161**(1314), 1943–1949 (2013)
7. Brešar, B., Kardoš, F., Katrenič, J., Semanišin, G.: Minimum k-path vertex cover. Discrete Appl. Math. **159**(12), 1189–1195 (2011)
8. Calinescu, G., Karloff, H., Rabani, Y.: Approximation algorithms for the 0-extension problem. SIAM J. Comput. **34**(2), 358–372 (2005)
9. Camby, E.: Connecting hitting sets and hitting paths in graphs. Doctoral Thesis (2015)
10. Chitnis, R., Hajiaghayi, M., Kortsarz, G.: Fixed-parameter and approximation algorithms: a new look. In: Gutin G., Szeider S. (eds.) Parameterized and Exact Computation. IPEC 2013. Lecture Notes in Computer Science, vol. 8246, pp. 110–122. Springer, Cham (2013)
11. Dinur, I., Guruswami, V., Khot, S., Regev, O.: A new multilayered PCP and the hardness of hypergraph vertex cover. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, STOC'03, pp. 595–601 (2003)
12. Drange, P.G., Dregi, M., van 't Hof, P.: On the computational complexity of vertex integrity and component order connectivity. Algorithmica **76**, 1181–1202 (2016). https://doi.org/10.1007/s00453-016-0127-x
13. Even, G., Naor, J., Rao, S., Schieber, B.: Fast approximate graph partitioning algorithms. SIAM J. Comput. **28**(6), 2187–2214 (1999)
14. Even, G., Naor, J., Rao, S., Schieber, B.: Divide-and-conquer approximation algorithms via spreading metrics. J. ACM **47**(4), 585–616 (2000)
15. Feige, U., Hajiaghayi, M., Lee, J.R.: Improved approximation algorithms for minimum weight vertex separators. SIAM J. Comput. **38**(2), 629–657 (2008)
16. Fomin, F.V., Lokshtanov, D., Misra, N., Saurabh, S.: Planar F-deletion: approximation, kernelization and optimal FPT algorithms. In: 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 470–479. IEEE (2012)
17. Fomin, F.V., Lokshtanov, D., Saurabh, S.: Efficient computation of representative sets with applications in parameterized and exact algorithms. In: Proceedings of the 25th Annual ACM–SIAM Symposium on Discrete Algorithms, pp. 142–151. Society for Industrial and Applied Mathematics (2014)
18. Garg, N., Vazirani, V.V., Yannakakis, M.: Approximate max-flow min-(multi) cut theorems and their applications. SIAM J. Comput. **25**(2), 235–251 (1996)
19. Giannopoulou, A.C., Pilipczuk, M., Thilikos, D.M., Raymond, J.F., Wrochna, M.: Linear kernels for edge deletion problems to immersion-closed graph classes. In: Proceedings of the 44th International Conference on Automata, Languages and Programming, ICALP'17 (2017)

20. Gross, D., Heinig, M., Iswara, L., Kazmierczak, L.W., Luttrell, K., Saccoman, J.T., Suffel, C.: A survey of component order connectivity models of graph theoretic networks. SWEAS Trans. Math. **12**(9), 895–910 (2013)
21. Guruswami, V., Lee, E.: Inapproximability of H-transversal/packing. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015), Leibniz International Proceedings in Informatics (LIPIcs), vol. 40, pp. 284–304 (2015)
22. Guruswami, V., Sachdeva, S., Saket, R.: Inapproximability of minimum vertex cover on k-uniform k-partite hypergraphs. SIAM J. Discrete Math. **29**(1), 36–58 (2015)
23. Hajiaghayi, M.T., Jain, K.: The prize-collecting generalized Steiner tree problem via a new approach of primal-dual schema. In: Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithm, pp. 631–640. Society for Industrial and Applied Mathematics (2006)
24. Katrenič, J.: A faster FPT algorithm for 3-path vertex cover. Inf. Process. Lett. **116**(4), 273–278 (2016)
25. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\epsilon$. J. Comput. Syst. Sci. **74**(3), 335–349 (2008)
26. Krauthgamer, R., Naor, J.S., Schwartz, R.: Partitioning graphs into balanced components. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 942–949. Society for Industrial and Applied Mathematics (2009)
27. Louis, A., Raghavendra, P., Vempala, S.: The complexity of approximating vertex expansion. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), pp. 360–369. IEEE (2013)
28. Lovász, L.: On minimax theorems of combinatorics. Doctoral Thesis, Mathematiki Lapok **26**, 209–264 (1975)
29. Manurangsi, P.: Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, pp. 954–961. ACM (2017)
30. Marx, D.: Parameterized complexity and approximation algorithms. Comput. J. **51**(1), 60–78 (2008)
31. Orlovich, Y., Dolgui, A., Finke, G., Gordon, V., Werner, F.: The complexity of dissociation set problems in graphs. Discrete Appl. Math. **159**(13), 1352–1366 (2011)
32. Papadimitriou, C.H., Yannakakis, M.: The complexity of restricted spanning tree problems. J. ACM **29**(2), 285–309 (1982)
33. Racke, H.: Optimal hierarchical decompositions for congestion minimization in networks. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC'08, pp. 255–264. ACM, New York (2008)
34. Trevisan, L.: Graph partitioning, expanders and spectral methods. In: UC Berkeley CS294 (2016). https://people.eecs.berkeley.edu/~luca/expanders2016/index.html
35. Tu, J., Zhou, W.: A factor 2 approximation algorithm for the vertex cover P_3 problem. Inf. Process. Lett. **111**(14), 683–686 (2011)
36. Yannakakis, M.: Node-deletion problems on bipartite graphs. SIAM J. Comput. **10**(2), 310–327 (1981)