# Partitioning a Graph into Small Pieces with Applications to Path Transversal[*]

Euiwoong Lee[†]

**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 at most $k$ vertices. Our primary focus is on the case where $k$ is either a constant or a slowly growing function of $n$ (e.g. $O(\log n)$ or $n^{o(1)}$). Our problems can be interpreted as a special case of three general classes of problems that have been studied separately (balanced graph partitioning, Hypergraph Vertex Cover (HVC), and fixed parameter tractability (FPT)).

Our main result is an $O(\log k)$-approximation algorithm for $k$-Vertex Separator that runs in time $2^{O(k)}n^{O(1)}$, and an $O(\log k)$-approximation algorithm for $k$-Edge Separator that runs in time $n^{O(1)}$. Our result on $k$-Edge Separator improves the best previous graph partitioning algorithm [24] for small $k$. Our result on $k$-Vertex Separator improves the simple $(k+1)$-approximation from HVC [3]. When $\mathsf{OPT} > k$, the running time $2^{O(k)}n^{O(1)}$ is faster than the lower bound $k^{\Omega(\mathsf{OPT})}n^{\Omega(1)}$ for exact algorithms assuming the Exponential Time Hypothesis [12]. While the running time of $2^{O(k)}n^{O(1)}$ for $k$-Vertex Separator seems unsatisfactory, we show that the superpolynomial dependence on $k$ may be needed to achieve a polylogarithmic approximation ratio, based on hardness of *Densest $k$-Subgraph.*

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 $k$. With additional ideas from FPT algorithms and graph theory, we present an $O(\log k)$-approximation algorithm for $k$-Path Transversal that runs in time $2^{O(k^3 \log k)}n^{O(1)}$. Previously, the existence of even $(1-\delta)k$-approximation algorithm for fixed $\delta > 0$ was open [9].

[*]Supported by the Samsung Scholarship, the Simons Award for Graduate Students in TCS, and Venkat Guruswami's NSF CCF-1115525. Part of this work was done when the author was an intern at Microsoft Research.

[†]Carnegie Mellon University

## 1 Introduction.

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 at most $k$ 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 at most $k$ vertices.

**Goal**: Minimize $|S|$.

These two problems 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.** 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. In this context, 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

[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 ($l = \frac{n}{k}$).

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) [31] 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. [24] 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 the Integer 3-Partition problem in graphs, and hard instances contain disjoint cliques of size at most $\frac{n}{l}$. Even et al. [13] defined a similar problem called *$\rho$-Separator*, which is exactly our $k$-Edge Separator with $\rho = \frac{k}{n}$. They "believe that the definition of $\rho$-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 $O(\frac{1+\epsilon}{\epsilon}\log n) \cdot \mathsf{OPT}$ edges to make sure that each component has size $(1+\epsilon)\rho n$ for any $\epsilon > 0$, which is improved to $O(\frac{1+\epsilon}{\epsilon}\sqrt{\log(1/\epsilon\rho)\log n}) \cdot \mathsf{OPT}$ by Krauthgamer et al. [24]. The previous algorithms' primary focus is when $\rho$ is a constant (so that $k = \Omega(n)$), and their performance deteriorates when $k$ is small. In particular, when $k = O(n^{1-\epsilon})$ and $\rho = O(n^{-\epsilon})$ for some $\epsilon > 0$, the best guarantee from the above line of work gives an $O(\log n)$-bicriteria approximation algorithm.

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 [25].

**Hypergraph Vertex Cover.** $k$-Vertex Separator for small values of $k$ has been actively studied. 1-Vertex Separator is the famous *Vertex Cover* problem. When $k = 2$, Papadimitriou and Yannakakis [34, 30] defined the *dissociation number* to be $n$ minus the optimum of 2-Vertex Separator in the context of certain constrained spanning tree problems, which have been studied independently from the graph partitioning literature (see [29] for a survey).

A simple $(k+1)$-approximation for $k$-Vertex Separator can be achieved by viewing them as a special case of *$(k+1)$-Hypergraph Vertex Cover ($(k+1)$-HVC)*. Given a graph $G = (V, E_G)$, we construct a hypergraph $H = (V, E_H)$ where $E_H$ contains every set of $k+1$ vertices $\{v_1, \ldots, v_{k+1}\} \subseteq V$ that induces a single connected component. This reduction is complete and sound because a subset $S \subseteq V$ intersects every hyperedge in $E_H$ if and only if $G|_{V \setminus S}$ has no connected component of size at least $k+1$. Since $(k+1)$-Hypergraph Vertex Cover admits a trivial $(k+1)$-approximation (e.g., take any hyperedge $e$ not intersecting $S$ and let $S \leftarrow S \cup e$), we get a $(k+1)$-true approximation for $k$-Vertex Separator. This was observed in the work of Ben-Ameur et al. [3].

Approximating $k$-HVC better than the trivial factor $k$ (resp. $k-1$) will refute the Unique Games Conjecture (resp. $\mathbf{P} \neq \mathbf{NP}$) [23, 11], 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 [26] gave a $\frac{k}{2}$-approximation algorithm that is shown to be tight under the Unique Games Conjecture [19] (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 [18] 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.

$k$-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 $k+1$ 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 $k+1$ vertices).

**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 to the survey by Gross et al. [17] for more background.

Let $\mathsf{OPT}$ be the optimal value. For small values of $k$ and $\mathsf{OPT}$, the complexity of exact algorithms has been studied in terms of their fixed parameter tractability (FPT). 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 $\mathsf{OPT}$. They complemented their result by showing that the problem is $\mathbf{W}[1]$-hard when parameterized by $\mathsf{OPT}$ or $k$. They also showed that any exact algorithm that runs in time $k^{o(\mathsf{OPT})}n^{O(1)}$ will refute the Exponential Time Hypothesis.

Given an instance of a problem with a parameter $\kappa$, 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 [27] and the recent work of Chitnis et al. [10]. For $k$-Vertex Separator, the simple $(k+1)$-approximation runs in polynomial time regardless of $\mathsf{OPT}$ and $k$, but any exact algorithm requires both $\mathsf{OPT}$ and $k$ to be parameterized. It is an interesting question whether significantly improved approximation is possible when only one of them is parameterized.

## 1.1 Our Results and Applications.

Our main result is 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 $G|_{S \setminus V}$ has at most $bk$ vertices and (2) $|S|$ is at most $c$ times the optimum of $k$-Vertex Separator.

Theorem 1.1. *For any $\epsilon \in (0, 1/4]$, there is a polynomial time $(\frac{1}{1-2\epsilon}, O(\frac{\log k}{\epsilon}))$-bicriteria approximation algorithm for $k$-Vertex Separator.*

Setting $\epsilon = \frac{1}{4}$ and running the algorithm yields $S \subseteq V$ with $|S| \leq O(\log k) \cdot \mathsf{OPT}$ such that each component in $G|_{V \setminus S}$ has at most $2k$ vertices. Performing an exhaustive search in each connected component yields the following true approximation algorithm whose running time depends exponentially only on $k$.

Corollary 1.1. *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 $(k+1)$-approximation from $k$-HVC. 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 Exponential Time Hypothesis [12].

The natural question is whether superpolynomial dependence on $k$ is necessary to achieve *true* $O(\log k)$-approximation. The following theorem proves hardness of $k$-Vertex Separator based on Densest $k$-Subgraph. In particular, a polynomial time $O(\log k)$-approximation algorithm for $k$-Vertex Separator will imply $O(\log^2 n)$-approximation algorithm for Densest $k$-Subgraph. Given that the best approximation algorithm achieves $\approx O(n^{1/4})$-approximation [4] and $n^{\Omega(1)}$-rounds of the Sum-of-Squares hierarchy have a gap at least $n^{\Omega(1)}$ [5], such a result seems unlikely or will be considered as a breakthrough.

Theorem 1.2. *If there is a polynomial time $f$-approximation algorithm for $k$-Vertex Separator, then there is a polynomial time $2f^2$-approximation algorithm for Densest $k$-Subgraph.*

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

When $k = n^{o(1)}$ so that $\rho = n^{-(1-o(1))}$, our algorithm outperforms the previous best approximation algorithm for $\rho$-separator [24, 13].[2]

While most graph partitioning algorithms deal with the edge version, we focus on the vertex version because (1) it exhibits richer connections to $k$-HVC and FPT as mentioned, (2) usually the vertex version is considered to be harder in the graph partitioning literature. We present the algorithm for the edge version in Appendix B.

**$k$-Path Transversal.** Let $l(G)$ be the length of the longest path of $G$ including both endpoints (e.g., length of a single edge is 2). Given a graph $G = (V, E)$ and $k \in \mathbb{N}$, *$k$-Path Transversal* asks to find the smallest subset $S \subseteq V$ such that $l(G|_{V \setminus S}) < k$. Finding a path of length $k$ has played a central role in devopment of FPT algorithms — it is NP-hard to do for general $k$, but there are various algorithms that run in time $2^{O(k)} n^{O(1)}$ using color coding or algebraic algorithms.

$k$-Path Transversal is motivated by applications in transportation / wireless sensor networks, and has also been actively studied as $k$-Path Vertex Cover or $P_k$-Hitting Set [32, 7, 6, 9, 21] in terms of their approximability and fixed parameter tractability. Tu and Zhou [32] gave a 2-approximation algorithm for 3-Path Transversal. Camby [9] recently gave a 3-approximation algorithm for 4-Path Transversal. In the same doctoral thesis, Camby [9] asked whether we can get $(1-\delta)k$-approximation for $k$-Path Transversal for a general $k$ and a universal constant $\delta > 0$. We show that it admits $O(\log k)$-approximation in FPT time. Note that the superpolynomial dependence on $k$ is necessary for any approximation from NP-hardness of finding a $k$-path.

Theorem 1.4. *There is an $O(\log k)$-approximation algorithm for $k$-Path Transversal that runs in time $2^{O(k^3 \log k)} n^{O(1)}$.*

## 2 Techniques.

Our algorithms for $k$-Vertex Separator and $k$-Edge Separator consist of the following three steps. We

[2] Both papers only present a bicriteria approximation algorithm, but they can be combined with our final *cleanup* step to achieve true approximation by *adding* $O(\log k)$ to the approximation ratio. See Appendix B.

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 $\rho$-separator [13].[3] They assign lengths to vertices such that any subset $S$ of vertices with $|S| > k$ that induce a single connected component are *spread apart*.

Given lengths $x_v$ to each vertex $v \in V$, define $d_{u,v}$ to be the length of the shortest 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 $G|_{V \setminus S}$ 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 at most $k$.

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 $\sum_u f_{v,u} \leq k$ 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 at most $k$.

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 $\epsilon \in (0, \frac{1}{4}]$, the number of vertices that have distance at most $\epsilon$ from $v$ can be at most $\frac{k}{1-2\epsilon}$. 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. [24] further strengthened the above spreading metrics by requiring that they also form an $\ell_2^2$ metric, and transformed them to an $\ell_2$ metric. This black-box transformation of an $n$-points $\ell_2^2$ 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. [16] 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 $O(\log(\frac{\mathsf{vol}(v,\frac{1}{2})}{\mathsf{vol}(v,0)}))$. To make $\mathsf{vol}(v, 0)$ nonzero, a *seed value* of $\epsilon \cdot \mathsf{OPT}$ 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 $\Omega(\epsilon n \mathsf{OPT})$, forcing $\epsilon$ 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 [16] 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 the superpolynomial dependence on $k$ may be necessary.

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 Min Bisection [31] to each component to make sure that each component has at most $k$ vertices. The existence of a *true approximation* for Min 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 Min Bisection [15], but our hardness result for the vertex version suggests that this algorithm is not likely to be applicable. While Min 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.

## 3 Algorithm for $k$-Vertex Separator.

**3.1 Spreading Metrics.** Our relaxation is close to *spreading metrics* used for $\rho$-separator [13]. While their

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

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. 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.

$$\begin{array}{llll}
 & \text{minimize} & \sum_{v \in V} x_v & \\
(3.1) & \text{s.t.} & d_{u,v} \le \min_{P \in \mathcal{P}_{u,v}} d(P) & \forall (u,v) \in V \times V \\
 & & f_{u,v} \ge 1 - d_{u,v} & \forall (u,v) \in V \times V \\
 & & f_{u,v} \ge 0 & \forall (u,v) \in V \times V \\
(3.2) & & \sum_{u \in V} f_{v,u} \le k & \forall v \in V \\
(3.3) & & x_v \ge 0 & \forall v \in V
\end{array}$$

(3.1) can be formally written as

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

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 $G|_{V \setminus S}$ has at most $k$ vertices, the following is a feasible solution with $\sum_v x_v = |S|$.

- $x_v = 1$ 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 $G|_{V \setminus S}$. 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 $d_{u,v} \le \min_{P \in \mathcal{P}_{u,v}} d(P)$, 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 (3.2). 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 small number of vertices, proving Theorem 1.1. 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 $\epsilon \in (0, \frac{1}{4}]$. 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 $x_v \ge \epsilon$. This simple step is crucial in bounding the size of the ball around each vertex. It removes at most $\frac{\mathsf{OPT}}{\epsilon}$ vertices. Let $V' := V \setminus \{v : x_v \ge \epsilon\}$, and $G' = (V', E')$ be the subgraph of $G$ induced by $V'$. 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, $d'_{u,v} \ge d_{u,v}$ and $f'_{u,v} \le f_{u,v}$ for all $(u,v) \in V' \times V'$.

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

- Pick $X \in [\epsilon/2, \epsilon]$ uniformly at random.

- Choose a permutation $\pi : V' \mapsto V'$ uniformly at random.

- Consider the vertices one by one, in the order given by $\pi$. 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 $d'_{w,v} - x_v \le X \le d'_{w,v}$, we remove $v$ when it was neither removed nor *disconnected* previously.

  – The vertices in $\{v : d'_{w,v} < X\}$ are now disconnected from the rest of the graph. Say these vertices are *disconnected*.

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

LEMMA 3.1. *For each vertex $w$, $|B(w)| \le \frac{k}{1-2\epsilon}$.*

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

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

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

$$\sum_{u \in V} f_{w,u} \geq \sum_{u \in B(w)} f_{w,u} \geq (1-2\epsilon)|B(w)| > k,$$

contradicting (3.2) of our LP.

Note that at the end of the algorithm, every 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 $\{v : d'_{w,v} < X\}$ for some $w \in V'$ and $X \leq \epsilon$, which is a subset of $B(w)$. Therefore, each connected component has at most $\frac{k}{1-2\epsilon}$ vertices. We finally analyze the probability that a vertex $v$ is removed.

Lemma 3.2. *The probability that $v \in V'$ is removed is at most $O(\frac{\log k}{\epsilon}) \cdot x_v$.*

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

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

Let $W = \{w_1, \ldots, w_p\}$ be such vertices such that $d'_{v,w_1} \leq \cdots \leq d'_{v,w_p} \leq 2\epsilon$. By Lemma 3.1, $p \leq \frac{k}{1-2\epsilon}$.

Fix $i$ and consider the event that $v$ is removed when $w_i$ is considered. This happens only if $d'_{v,w_i} - x_v \leq X \leq d'_{v,w_i}$. For fixed such $X$, a crucial observation is that if $w_j$ with $j < i$ is considered before $w_i$, since $d'_{v,w_j} - x_v \leq X$, $v$ 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

$$\begin{aligned} & \Pr[v \text{ is removed}] \\ & = \sum_{i=1}^{p} \Pr[v \text{ is removed when } w_i \text{ is considered}] \\ & \leq \sum_{i=1}^{p} \Pr[X \in [d'_{v,w_i} - x_v, d'_{v,w_i}] \text{ and} \\ & \qquad\qquad w_i \text{ comes before } w_1, \ldots, w_{i-1} \text{ in } \pi] \\ & \leq \sum_{i=1}^{p} \frac{2x_v}{\epsilon i} = x_v \cdot O(\frac{\log p}{\epsilon}) = x_v \cdot O(\frac{\log k}{\epsilon}). \end{aligned}$$

Therefore, the low-diameter decomposition removes at most $O(\frac{\log k}{\epsilon}) \cdot \sum_v x_v \leq O(\frac{\log k}{\epsilon}) \cdot \mathsf{OPT}$ vertices so that each resulting connected component has at most $\frac{k}{1-2\epsilon}$ vertices. This gives a bicriteria approximation algorithm that runs in time *poly*$(n, k)$, proving Theorem 1.1.

## 4 $k$-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 $G|_{V \setminus S}$ (denoted by $l(G|_{V \setminus S})$) 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.

Let $\mathcal{P}_k$ be the set of all simple paths of length $k$. Our algorithm starts by solving the following naive LP.

$$\begin{aligned} & \text{minimize} && \sum_{v \in V} x_v && \\ (4.4) \quad & \text{s.t.} && \sum_{i=1}^{k} x_{v_i} \geq 1 && \forall P = (v_1, \ldots, v_k) \in \mathcal{P}_k \\ & && x \geq 0 && \forall v \in V \times V \end{aligned}$$

When $G$ is a clique with $n$ vertices, any feasible solution needs to remove at least $n-k+1$ vertices while the above LP has the optimum at most $\frac{n}{k}$ by giving $\frac{1}{k}$ to every $x_v$. Therefore, it has an integrality gap close to $k$, but our algorithm bypasses this gap.

Lemma 4.1. *The above LP can be solved in time $2^{O(k)} n^{O(1)}$.*

*Proof.* Given the current solution $\{x_v\}_v$, we show how to check (4.4) in FPT time, so that the LP can be solved efficiently via the ellipsoid algorithm. In particular, it suffices to compute $\min_{P=(v_1,\ldots,v_k) \in \mathcal{P}_k} \sum_{i=1}^{k} x_{v_i}$. Our algorithm is a simple variant of an algorithm for the $k$-Path problem. Our presentation follows Williams [33].

Call a set of functions $F = \{f_i\}_i$ with $f_i : V \mapsto [k]$ a *$k$-perfect hash family* if for any subset $S \subseteq V$ with $|S| = k$, there exists $f_i \in F$ such that $f_i(S) = [k]$ (say that $S$ is *shattered* by $f_i$). Naor et al. [28] show that such an efficiently computable family $F$ exists with $|F| = 2^{O(k)} \log n$.

For each $f_i \in F$, we compute $\min_{P=(v_1,\ldots,v_k)} \sum_{i=1}^{k} x_{v_i}$, where $P$ ranges over a set of $k$-paths shattered by $f_i$. This can be done in time $2^{O(k)} n^{O(1)}$ using dynamic programming. For $v \in V$ and $S \subseteq [k]$ such that $f_i(v) \in S$, let $T[v, S]$ be the minimum weighted length of a path that (1) has exactly $|S|$ vertices, one from $f_i^{-1}(j)$ for each $j \in S$ and (2) ends at $v$. There are at most $2^k n$ entries and computing each entry takes time at most $kn$, so the entire $T$ can be computed in time $k2^k n^2$. One of $\{T[v,[k]]\}_{v \in V}$ gives the desired minimum.

Let $P^* = (v_1^*, \ldots, v_k^*)$ be the path that minimizes $\min_{P=(v_1,\ldots,v_k)\in\mathcal{P}_k} \sum_{i=1}^k x_{v_i}$. There must be $f_i \in F$ that shatters $P^*$, and the above dynamic programming algorithm for $f_i$ finds $P^*$. The separation oracle runs in time $|F| \cdot (k2^k n^2) = 2^{O(k)} n^{O(1)}$. Our LP has only $n$ variables, so the total LP runs in time $2^{O(k)} n^{O(1)}$.

Solve the above LP to get an optimal solution $\{x_v\}_{v \in V}$. Let $\mathsf{FRAC} := \sum_v x_v$. Call a vertex $v \in V$ *red* if $x_v \geq \frac{1}{k}$. Let $R$ be the set of red vertices. One simple but crucial observation is that every $k$-path must contain at least one red vertex, since all non-red vertices have $x_v < \frac{1}{k}$.

Let $S^*$ be the optimal solution of $k$-Path Transversal. Let $V^* := V \setminus S^*, R^* := R \setminus S^*$ and $G^* = G|_{V \setminus S^*}$. The result for $k$-Path Transversal requires the following lemma.

LEMMA 4.2. *There exists $S' \subseteq V^*$ with $|S'| \leq \frac{|R^*|}{k}$ vertices so that in the induced subgraph $G^*_{V^* \setminus S'}$, 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 following simple observation.

FACT 4.1. *In a connected component $C$, any two longest paths should intersect.*

*Proof.* Let $P_1 = (v_1, \ldots, v_p)$ and $P_2 = (u_1, \ldots, u_p)$ be two vertex-disjoint longest paths in the same connected component. Since they are in the same component, there exist $i, j \in [k]$ and another path $P_3 = (v_i, w_1, \ldots, w_q, u_j)$ such that $w_1, \ldots, w_q$ are disjoint from $v$'s and $u$'s ($q$ may be 0). By reversing the order of $P_1$ or $P_2$, we can assume that $i, j \geq \frac{p+1}{2}$. Then $(v_1, \ldots, v_i, w_1, \ldots, w_q, u_j, \ldots, u_1)$ is a path with length at least $p+1$, contradicting the fact that $P_1$ and $P_2$ are longest paths.

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 $S^* \cup S'$. Its size is at most $\mathsf{OPT} + \frac{|R^*|}{k} \leq \mathsf{OPT} + \mathsf{FRAC} \leq 2\mathsf{OPT}$, since every red vertex has $x_v \geq \frac{1}{k}$, and each component of $G_{S^* \cup S'}$ has at most $k^3$ red vertices. We formally define the following generalization of $k$-Vertex Separator.

**$k$-Subset Vertex Separator**

**Input**: An undirected graph $G = (V, E)$, a subset $R \subseteq V$ 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 at most $k$ vertices from $R$.

**Goal**: Minimize $|S|$.

Even though it seems a nontrivial generalization of $k$-Vertex Separator, the analogous bicriteria approximation algorithm also exists. It is proved in Section C.

THEOREM 4.1. *For any $\epsilon \in (0, 1/4]$, there is a polynomial time $(\frac{1}{1-2\epsilon}, O(\frac{\log k}{\epsilon}))$-bicriteria approximation algorithm for $k$-Subset Vertex Separator.*

For $k$-Path Transversal, run the above bicriteria approximation algorithm for $k$-Subset Vertex Separator with $k \leftarrow k^3$ and $\epsilon \leftarrow \frac{1}{4}$. This returns a subset $S \subseteq V$ such that $|S| \leq O(\log k) \cdot \mathsf{OPT}$ and each connected component of $G_{V \setminus S}$ has at most $2k^3$ red vertices.

Now we solve 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 $1 \leq i \leq k$,
    - Remove $v_i$ from the graph and recurse.

Finding a path takes time $2^{O(k)} n^{O(1)}$. 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. Therefore, it runs in time $2^{O(k)}n^{O(1)} \cdot k^{2k^3} = 2^{O(k^3 \log k)}n^{O(1)}$. This proves Theorem 1.4.

**Acknowledgements.** We thank Anupam Gupta, Guru Guruganesh, Venkatesan Guruswami, Konstantin Makarychev, and Tselil Schramm for useful discussions. We are also grateful to anonymous reviewers for FOCS 2016 and SODA 2017 to suggest a way to improve our algorithms.

## References

[1] K. Andreev and H. Racke, *Balanced graph partitioning*, Theory of Computing Systems, 39 (2006), pp. 929–939.

[2] S. Arora, S. Rao, and U. Vazirani, *Expander flows, geometric embeddings and graph partitioning*, Journal of ACM, 56 (2009).

[3] W. Ben-Ameur, M.-A. Mohamed-Sidi, and J. Neto, *The k-separator problem: polyhedra, complexity and approximation results*, Journal of Combinatorial Optimization, 29 (2015), pp. 276–307.

[4] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan, *Detecting high log-densities: an $o(n^{1/4})$ approximation for densest k-subgraph*, in Proceedings of the forty-second ACM symposium on Theory of computing, ACM, 2010, pp. 201–210.

[5] A. Bhaskara, M. Charikar, A. Vijayaraghavan, V. Guruswami, and Y. Zhou, *Polynomial integrality gaps for strong sdp relaxations of densest k-subgraph*, in Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms, SIAM, 2012, pp. 388–405.

[6] B. Brear, M. Jakovac, J. Katreni, G. Semaniin, and A. Taranenko, *On the vertex k-path cover*, Discrete Applied Mathematics, 161 (2013), pp. 1943 – 1949.

[7] B. Brešar, F. Kardoš, J. Katrenič, and G. Semanišin, *Minimum k-path vertex cover*, Discrete Applied Mathematics, 159 (2011), pp. 1189–1195.

[8] G. Calinescu, H. Karloff, and Y. Rabani, *Approximation algorithms for the 0-extension problem*, SIAM Journal on Computing, 34 (2005), pp. 358–372.

[9] E. Camby, *Connecting hitting sets and hitting paths in graphs*, Doctoral Thesis, (2015).

[10] R. Chitnis, M. Hajiaghayi, and G. Kortsarz, *Fixed-parameter and approximation algorithms: A new look*, in Parameterized and Exact Computation, Springer, 2013, pp. 110–122.

[11] I. Dinur, V. Guruswami, S. Khot, and O. Regev, *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, 2003, pp. 595–601.

[12] P. G. Drange, M. S. Dregi, and P. vant Hof, *On the computational complexity of vertex integrity and component order connectivity*, in Algorithms and Computation, Springer, 2014, pp. 285–297.

[13] G. Even, J. Naor, S. Rao, and B. Schieber, *Fast approximate graph partitioning algorithms*, SIAM Journal on Computing, 28 (1999), pp. 2187–2214.

[14] G. Even, J. Naor, S. Rao, and B. Schieber, *Divide-and-conquer approximation algorithms via spreading metrics*, Journal of the ACM, 47 (2000), pp. 585–616.

[15] U. Feige, M. Hajiaghayi, and J. R. Lee, *Improved approximation algorithms for minimum weight vertex separators*, SIAM Journal on Computing, 38 (2008), pp. 629–657.

[16] N. Garg, V. V. Vazirani, and M. Yannakakis, *Approximate max-flow min-(multi) cut theorems and their applications*, SIAM Journal on Computing, 25 (1996), pp. 235–251.

[17] D. Gross, M. Heinig, L. Iswara, L. W. Kazmierczak, K. Luttrell, J. T. Saccoman, and C. Suffel, *A survey of component order connectivity models of graph theoretic networks*, SWEAS Trans. Math, 12 (2013), pp. 895–910.

[18] V. Guruswami and E. Lee, *Inapproximability of H-Transversal/Packing*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015), N. Garg, K. Jansen, A. Rao, and J. D. P. Rolim, eds., vol. 40 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2015, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 284–304.

[19] V. Guruswami, S. Sachdeva, and R. Saket, *Inapproximability of minimum vertex cover on k-uniform k-partite hypergraphs*, SIAM Journal on Discrete Mathematics, 29 (2015), pp. 36–58.

[20] M. T. Hajiaghayi and K. Jain, *The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema*, in Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, Society for Industrial and Applied Mathematics, 2006, pp. 631–640.

[21] J. Katrenič, *A faster FPT algorithm for 3-path vertex cover*, Information Processing Letters, 116 (2016), pp. 273–278.

[22] S. Khot, *Ruling out ptas for graph minbisection, dense ksubgraph, and bipartite clique*, SIAM Journal on Computing, 36 (2006), pp. 1025–1071.

[23] S. Khot and O. Regev, *Vertex cover might be hard to approximate to within $2 - \epsilon$*, Journal of Computer and System Sciences, 74 (2008), pp. 335–349.

[24] R. Krauthgamer, J. S. Naor, and R. Schwartz, *Partitioning graphs into balanced components*, in Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2009, pp. 942–949.

[25] A. Louis, P. Raghavendra, and S. Vempala, *The complexity of approximating vertex expansion*, in Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on, IEEE, 2013, pp. 360–369.

[26] L. Lovász, *On minimax theorems of combinatorics*, Doctoral Thesis, Mathematiki Lapok, 26 (1975), pp. 209–264.

[27] D. Marx, *Parameterized complexity and approximation algorithms*, The Computer Journal, 51 (2008), pp. 60–78.

[28] M. Naor, L. J. Schulman, and A. Srinivasan, *Splitters and near-optimal derandomization*, in Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on, IEEE, 1995, pp. 182–191.

[29] Y. Orlovich, A. Dolgui, G. Finke, V. Gordon, and F. Werner, *The complexity of dissociation set problems in graphs*, Discrete Applied Mathematics, 159 (2011), pp. 1352–1366.

[30] C. H. Papadimitriou and M. Yannakakis, *The complexity of restricted spanning tree problems*, Journal of the ACM, 29 (1982), pp. 285–309.

[31] H. Racke, *Optimal hierarchical decompositions for congestion minimization in networks*, in Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC '08, New York, NY, USA, 2008, ACM, pp. 255–264.

[32] J. Tu and W. Zhou, *A factor 2 approximation algorithm for the vertex cover p3 problem*, Information Processing Letters, 111 (2011), pp. 683–686.

[33] R. Williams, *Stanford CS266: Parameterized algorithms and complexity*, Lecture 4, (2013).

[34] M. Yannakakis, *Node-deletion problems on bipartite graphs*, SIAM Journal on Computing, 10 (1981), pp. 310–327.

## A 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 the Densest $k$-Subgraph problem, proving Theorem 1.2. 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 $G|_S$. It is one of the notorious problems in approximation algorithms. The current best approximation algorithm achieves $\approx O(n^{1/4})$-approximation [4]. While only PTAS is ruled out assuming $\mathbf{NP} \not\subseteq \cap_{\epsilon>0}\mathbf{BPTIME}(2^{n^{\epsilon}})$ [22], there are strong gap instances for Sum-of-Squares hierarchies of convex relaxations ($n^{\Omega(1)}$ gap for $n^{\Omega(1)}$ rounds) [5], so having a $polylog(n)$-approximation algorithm for Densest $k$-Subgraph seems unlikely or will lead to a breakthrough. Therefore, it may be the case that achieving $O(\log k)$-approximation for $k$-Vertex Separator requires superpolynomial dependence on $k$ in the running time.

Our reduction is close to that of Drange et al. [12] who reduced Clique to $k$-Vertex Separator to prove $\mathbf{W}[1]$-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 [20] proved the following theorem, relating their approximation ratios.

Theorem A.1. ([20]) *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 A.1. *Every instance 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 $S$ such that $G'_{V' \setminus S}$ has each connected component with at most $k$ vertices. 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 $G'|_{V' \setminus S}$. 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 $u$ to $S$ decreases the size of the connected component of $u$ 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 $S \subseteq V$, $G'|_{V' \setminus S}$ has the following connected components.

- One component $(V \setminus S) \cup \{e_i : e = (u,v) \in E, \{u,v\} \not\subseteq S, i \in [M]\}$. 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 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 $|V \setminus S| = n - |T|$ and $|\{(u,v) \in E : \{u,v\} \not\subseteq T\}| = m - l$, In $G'|_{V' \setminus S}$, the giant component will have cardinality

$$\begin{aligned} &n - |T| + M(m-l) \leq n - |T| + M(m-k) \\ &\leq n + M(m-k) = |V'| - Mk = k'. \end{aligned}$$

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

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

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 1.2.

## B Algorithm for $k$-Edge Separator.

We present an $O(\log k)$-true approximation algorithm for $k$-Edge Separator, proving Theorem 1.3. Except the cleanup step, the algorithm is almost identical to that of $k$-Vertex Separator.

**B.1 Spreading Metrics.** Our relaxation for the edge version is very close to that of the vertex version. It has the following variables.

- $x_e$ for $e \in E$: It indicates whether $e$ is removed or not.
- $d_{u,v}$ for $(u,v) \in V \times V$: Given $\{x_e\}_{e \in E}$ 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 $P = (u_0 := u, u_1, \dots, u_p := v) \in \mathcal{P}_{u,v}$, let $d(P) = x_{u_0,u_1} + \cdots + x_{u_{p-1},u_p}$. Formally, we want
$$d_{u,v} = \min_{P \in \mathcal{P}_{u,v}} d(P).$$
Note that $d_{u,v} = d_{v,u}$ and $d_{u,u} = 0$.
- $f_{u,v}$ 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.

$$\begin{array}{llll} & \text{minimize} & \sum_{e \in E} x_e & \\ \text{(B.1)} & \text{s.t.} & d_{u,v} \leq \min_{P \in \mathcal{P}_{u,v}} d(P) & \forall (u,v) \in V \times V \\ & & f_{u,v} \geq 1 - d_{u,v} & \forall (u,v) \in V \times V \\ & & f_{u,v} \geq 0 & \forall (u,v) \in V \times V \\ \text{(B.2)} & & \sum_{u \in V} f_{v,u} \leq k & \forall v \in V \\ \text{(B.3)} & & x_e \geq 0 & \forall e \in E \end{array}$$

(B.1) can be formally written as

$$\begin{array}{ll} d_{u,u} = 0 & \forall u \in V \\ d_{u,w} \leq d_{u,v} + x_{v,w} & \forall (u,v) \in V \times V, (v,w) \in E \end{array}$$

Therefore, the size of our LP is polynomial in $n$. It is easy to verify that our LP is a relaxation — given a subset $S \subseteq E$ such that each connected component of $(V, E \setminus S)$ has at most $k$ vertices, the following is a feasible solution with $\sum_e x_e = |S|$.

- $x_e = 1$ if $e \in S$. $x_e = 0$ 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 $(V, E \setminus S)$. Otherwise $f_{u,v} = 0$.

Fix an optimal solution $\{x_e\}_e, \{d_{u,v}, f_{u,v}\}_{u,v}$ 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 $f_{u,v}$. For the subsequent sections, we assume that $d_{u,v} = \min_{P \in \mathcal{P}_{u,v}} d(P)$ for all $u, v$.

**B.2 Low-diameter Decomposition.** Fix $\epsilon \in (0, \frac{1}{2}]$. Given an optimal solution $\{x_e\}_{e \in E}$ and $\{d_{u,v}\}_{u,v \in V \times V}$ to the above LP, our low-diameter decomposition removes at most $O(\frac{\log k}{\epsilon}) \cdot \sum_e x_e$ edges so that each resulting connected component has at most $\frac{k}{1-\epsilon}$ vertices. It proceeds as follows.

- Pick $X \in [\epsilon/2, \epsilon]$ uniformly at random.
- Choose a permutation $\pi : V \mapsto V$ uniformly at random.
- Consider the vertices one by one, in the order given by $\pi$. Let $w$ be the considered vertex (we consider every vertex whether it was previously disconnected or not).
  - Let $W \leftarrow \emptyset$.
  - For each vertex $v \in V$ with $d_{w,v} \le X$, if it is not disconnected yet, add it to $W$.
  - *Disconnect* $W$ from the rest of the graph (i.e., remove every edge that has exactly one endpoint in $W$).

For each vertex $w$, let $B(w) := \{v : d_{w,v} \le \epsilon\}$. A simple averaging argument bounds $|B(w)|$.

Lemma B.1. *For each vertex $w$, $|B(w)| \le \frac{k}{1-\epsilon}$.*

*Proof.* Assume towards contradiction that $|B(w)| > \frac{k}{1-\epsilon}$. For all $u \in B(w)$, $f_{w,u} \ge 1 - d_{w,u} \ge 1 - \epsilon$. Furthermore, even for $u \notin B(w)$, our LP ensures that $f_{w,u} \ge 0$. Therefore,

$$\sum_{u \in V} f_{w,u} \ge \sum_{u \in B(w)} f_{w,u} \ge (1-\epsilon)|B(w)| > k,$$

contradicting (B.2) of our LP.

Note that at the end of the algorithm, every vertex is disconnected, since every $w \in V$ becomes disconnected after being considered. Moreover, each connected component is a subset of $\{v : d_{w,v} \le X\}$ for some $w \in V$ and $X \le \epsilon$, which is a subset of $B(w)$. Therefore, each connected component has at most $\frac{k}{1-\epsilon}$ vertices. We finally analyze the probability that an edge $e$ is removed.

Lemma B.2. *The probability that $e \in E$ is removed is at most $O(\frac{\log k}{\epsilon}) \cdot x_e$.*

*Proof.* Fix an edge $e = (u, v) \in E$. For a vertex $v \in W$, let $d^{\mathsf{near}}_{w,e} = \min(d_{w,u}, d_{w,v})$ and $d^{\mathsf{far}}_{w,e} = \max(d_{w,u}, d_{w,v})$. When $w \in V$ is considered, $e$ can be possibly removed only if $d^{\mathsf{near}}_{w,e} \le \epsilon \Rightarrow w \in B(v) \cup B(u)$. Let $W = \{w_1, \dots, w_p\}$ be such vertices such that $d^{\mathsf{near}}_{w_1,e} \le \cdots \le d^{\mathsf{near}}_{w_p,e} \le \epsilon$. By Lemma B.1, $p \le 2 \cdot \frac{k}{1-\epsilon}$.

Fix $i$ and consider the event that $e$ is removed when $w_i$ is considered. This happens only if $d^{\mathsf{near}}_{w_i,e} \le X \le d^{\mathsf{far}}_{w_i,e}$. For fixed such $X$, a crucial observation is that if $w_j$ with $j < i$ is considered before $w_i$, since $d^{\mathsf{near}}_{w_j,e} \le X$, $e$ will be either removed (exactly one of $u$ and $v$ is disconnected) or disconnected (both $u$ and $v$ are disconnected) when $w_j$ is considered. In particular, $e$ will not be removed by $w_i$. Given these observations, the probability that $e$ is removed is bounded by

$$\begin{aligned}
&\Pr[e \text{ is removed}] \\
&= \sum_{i=1}^{p} \Pr[e \text{ is removed when } w_i \text{ is considered}] \\
&\le \sum_{i=1}^{p} \Pr[X \in [d^{\mathsf{near}}_{v,w_i}, d^{\mathsf{far}}_{v,w_i}] \text{ and} \\
&\qquad\qquad w_i \text{ comes before } w_1, \dots, w_{i-1} \text{ in } \pi] \\
&\le \sum_{i=1}^{p} \frac{2x_e}{\epsilon i} = x_e \cdot O(\frac{\log p}{\epsilon}) = x_v \cdot O(\frac{\log k}{\epsilon}).
\end{aligned}$$

Therefore, the low-diameter decomposition removes at most $O(\frac{\log k}{\epsilon}) \cdot \sum_v x_v \le O(\frac{\log k}{\epsilon}) \cdot \mathsf{OPT}$ edges so that each resulting connected component has at most $\frac{k}{1-\epsilon}$ vertices.

To get true approximation, we use the algorithm for *Balanced b-Cut*. For an undirected graph $G = (V, E)$ with $n$ vertices and a real $b \in (0, 1/2]$, the Balanced $b$-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 [31] gave an $O(\log n)$-true approximation algorithm for Balanced $b$-cut.[4]

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

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

[4]His algorithm is originally stated for *Min 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 Balanced $b$-Cut. The author thanks to Anupam Gupta for this idea.

*Proof.* Any cut $(S, V \setminus S)$ feasible for $b$-Balanced Cut ensures that $\max(|S|, |V \setminus S|)$ is at most $(1-b)k' = k$, 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 $S_1, \dots, S_l$ (assume $k \geq |S_1| \geq \cdots \geq |S_l|$), if $l = 2$, $(S_1, S_2)$ is a feasible solution for $b$-Balanced Cut and we are done. If $l \geq 3$, merge $S_{l-1}, S_l$ into one set (one $S_i$ may contain multiple connected components). This reduces $l$ by 1, and since $|S_{l-1}| + |S_l| \leq \frac{2}{l} \cdot k' \leq \frac{2}{3}k' \leq k$, maintains the invariant that $|S_i| \leq k$ 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 at most $k$ vertices. This proves Theorem 1.3.

## C $k$-Subset Vertex Separator.

Given a graph $G = (V, E)$ and $k \in \mathbb{N}$. There is 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 $P = (u_0 := u, u_1, \dots, u_p := v) \in \mathcal{P}_{u,v}$, 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.

$$\begin{array}{llll} & \text{minimize} & \sum_{v \in V} x_v & \\ \text{(C.4)} & \text{s.t.} & d_{u,v} \leq \min_{P \in \mathcal{P}_{u,v}} d(P) & \forall (u,v) \in V \times V \\ & & f_{u,v} \geq 1 - d_{u,v} & \forall (u,v) \in V \times V \\ & & f_{u,v} \geq 0 & \forall (u,v) \in V \times V \\ \text{(C.5)} & & \sum_{u \in R} f_{v,u} \leq k & \forall v \in V \end{array}$$

(C.4) can be formally written as

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

The only change is that in (C.5), $f_{v,u}$ is summed over $u \in R$ instead of $u \in V$. It is clearly a relaxation.

Fix an optimal solution $\{x_v\}_v, \{d_{u,v}, f_{u,v}\}_{u,v}$ for the above LP. As usual, assume without loss of generality 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 $u, v$.

**C.1 Low-diameter Decomposition.** Fix $\epsilon \in (0, \frac{1}{4}]$. 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 $x_v \geq \epsilon$. It removes at most $\frac{\mathsf{OPT}}{\epsilon}$ vertices.

Let $V' := V \setminus \{v : x_v \geq \epsilon\}$, 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 at most $O(\frac{\log k}{\epsilon}) \cdot \sum_{v \in V'} x_v$ vertices so that each resulting connected component has at most $\frac{k}{1-2\epsilon}$ red vertices. It proceeds as follows.

- Pick $X \in [\epsilon/2, \epsilon]$ 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 $\pi$. 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 $d'_{w,v} - x_v \leq X \leq d'_{w,v}$, we remove $v$ when it was neither removed nor *disconnected* previously.
  - The vertices in $\{v : d'_{w,v} < X\}$ are now disconnected from the rest of the graph. Say these vertices are *disconnected*.

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

LEMMA C.1. *For each vertex $w \in V'$, $|B(w)| \leq \frac{k}{1-2\epsilon}$.*

*Proof.* Assume towards contradiction that $|B(w)| > \frac{k}{1-2\epsilon}$. For all $u \in B(w)$,
$$f_{w,u} \geq f'_{w,u} \geq 1 - d'_{w,u} \geq 1 - 2\epsilon.$$

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

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

contradicting (C.5) 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 $\{v : d'_{w,v} < X\}$ for some $w \in V'$ and $X \leq \epsilon$, which is a subset of $B(w)$. Therefore, each connected component has at most $\frac{k}{1-2\epsilon}$ red vertices. We finally analyze the probability that a vertex $v \in V'$ is removed.

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

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

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

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

Fix $i$ and consider the event that $v$ is removed when $w_i$ is considered. This happens only if $d'_{v,w_i} - x_v \leq X \leq d'_{v,w_i}$. For fixed such $X$, a crucial observation is that if $w_j$ with $j < i$ is considered before $w_i$, since $d'_{v,w_j} - x_v \leq X$, $v$ 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

$$\begin{aligned} & \Pr[v \text{ is removed}] \\ & = \sum_{i=1}^{p} \Pr[v \text{ is removed when } w_i \text{ is considered}] \\ & \leq \sum_{i=1}^{p} \Pr[X \in [d'_{v,w_i} - x_v, d'_{v,w_i}] \text{ and} \\ & \qquad\qquad w_i \text{ comes before } w_1, \ldots, w_{i-1} \text{ in } \pi] \\ & \leq \sum_{i=1}^{p} \frac{2x_v}{\epsilon i} = x_v \cdot O(\frac{\log p}{\epsilon}) = x_v \cdot O(\frac{\log k}{\epsilon}). \end{aligned}$$

Therefore, the low-diameter decomposition removes at most $O(\frac{\log k}{\epsilon}) \cdot \sum_v x_v \leq O(\frac{\log k}{\epsilon}) \cdot \mathsf{OPT}$ vertices so that each resulting connected component has at most $\frac{k}{1-2\epsilon}$ red vertices. This gives a bicriteria approximation algorithm that runs in time $poly(n, k)$, proving Theorem 4.1.