Near-Linear-Time, Optimal Vertex Cut Sparsifiers in Directed Acyclic Graphs

Zhiyang He
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Jason Li
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Magnus Wahlström
Royal Holloway, University of London, UK

Abstract

Let G be a graph and $S, T \subseteq V(G)$ be (possibly overlapping) sets of terminals, $|S| = |T| = k$. We are interested in computing a *vertex sparsifier* for terminal cuts in G, i.e., a graph H on a smallest possible number of vertices, where $S \cup T \subseteq V(H)$ and such that for every $A \subseteq S$ and $B \subseteq T$ the size of a minimum (A, B)-vertex cut is the same in G as in H. We assume that our graphs are unweighted and that terminals may be part of the min-cut. In previous work, Kratsch and Wahlström (FOCS 2012/JACM 2020) used connections to matroid theory to show that a vertex sparsifier H with $O(k^3)$ vertices can be computed in randomized polynomial time, even for arbitrary digraphs G. However, since then, no improvements on the size $O(k^3)$ have been shown.

In this paper, we draw inspiration from the renowned *Bollobás's Two-Families Theorem* in extremal combinatorics and introduce the use of total orderings into Kratsch and Wahlström's methods. This new perspective allows us to construct a sparsifier H of $\Theta(k^2)$ vertices for the case that G is a DAG. We also show how to compute H in time near-linear in the size of G, improving on the previous $O(n^{\omega+1})$. Furthermore, H recovers the *closest* min-cut in G for every partition (A, B), which was not previously known. Finally, we show that a sparsifier of size $\Omega(k^2)$ is required, both for DAGs and for undirected edge cuts.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Fixed parameter tractability; Mathematics of computing → Matroids and greedoids

Keywords and phrases graph theory, vertex sparsifier, representative family, matroid

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.52

Related Version *Full Version*: `https://arxiv.org/abs/2011.13485`

Funding *Jason Li*: Supported in part by NSF award CCF-1907820.

1 Introduction

Let $G = (V, E)$ be an unweighted, directed graph, and let $S, T \subset V$ be sets of terminals, not necessarily disjoint. In the vertex sparsifier problem, our goal is to construct a smaller graph H, called a *vertex sparsifier*, that preserves the cut structure of S, T in G. More precisely, H includes all vertices in S, T, and for all subsets $A \subseteq S$ and $B \subseteq T$, the size of the minimum vertex cut separating A and B is the same in G and H. Here, we allow the min-cut to contain vertices from A and B; for other notions of cut sparsifiers from the literature, see related work, below.

A result of Kratsch and Wahlström proved the first bound on the size of a vertex sparsifier that is polynomial in the number of terminals. When S, T have size k, the vertex sparsifier has $O(k^3)$ vertices. Kratsch and Wahlström's main insight is to phrase the problem in terms of constructing representative families on a certain matroid, after which they can appeal to the rich theory on representative families [19, 16]. Their result, also known as the *cut-covering lemma* in the areas of fixed-parameter tractability and kernelization, has led to many new algorithmic developments [13, 12, 9, 10]. Nevertheless, despite the recent surge in applications of the cut-covering lemma, the original bound of $O(k^3)$ has yet to be improved.

In this paper, we introduce the ordered version of the representative family method, and use it to give a sparsifier on $O(k^2)$ vertices in directed acyclic graphs. This matches lower bounds of $\Omega(k^2)$, which we present in Section 4. Furthermore, unlike Kratsch and Wahlström's result, our new algorithm runs in near-linear time in the size of the graph, and preserves all *closest* min-cuts between subsets of the terminals. In fact, the latter is an important ingredient in the improved running time; see discussion below. We expect that covering closest min-cuts may lead to further applications in the theory of kernelization. The central method we use is the following theorem.

▶ **Theorem 1.** *Suppose $\mathcal{F}$ is a family of subsets of $\mathbb{F}^d$ for some field $\mathbb{F}$ and for all $B \in \mathcal{F}$, $|B| = s$. Let*

$$\mathcal{A} = \{A \subseteq \mathbb{F}^d \mid |A| \leq r \textit{ and } \exists B \in \mathcal{F} \textit{ s.t. } A \uplus B \textit{ is linearly independent}\}.$$

Fix any ordering σ of $\mathcal{F}$, namely $\mathcal{F} = \{B_1, B_2, \dots, B_n\}$, and suppose $d \geq r + s$. Then there exists $\mathcal{B} \subseteq \mathcal{F}$, $\mathcal{B} = \{B_{i_1}, B_{i_2}, \dots, B_{i_m}\}$ where $i_1 < i_2 < \ldots < i_m$, such that

(a) *For all $A \in \mathcal{A}$, there exists $B_{i_k} \in \mathcal{B}$ where $A \uplus B_{i_k}$ is independent and for all $j \in [n], j > i_k$, $A \uplus B_j$ is dependent. Note that B_j is not necessarily in $\mathcal{B}$. Here $\uplus$ denotes disjoint union, which particularly implies that if A and B are not disjoint, then $A \uplus B$ is a multi-set and therefore dependent.*

(b) *$m \leq \binom{d}{s}$, and we can find $\mathcal{B}$ algorithmically using $O(\binom{d}{s} ns^{\omega} + \binom{d}{s}^{\omega-1} n)$ field operations over $\mathbb{F}$ (in particular, in a number of operations linear in $|\mathcal{F}|$).*

The condition that $A \uplus B_j$ is required to be dependent only for $j > i_k$, as opposed to $A \uplus B_j$ being dependent for every $j \neq i_k$, recalls the *skew* version of Bollobás's two-families theorem, proven by Frankl [8].

Technically, this version is equivalent to the weighted version of the representative family method shown by Fomin et al. [6]. In that version, every element $X \in \mathcal{F}$ has a weight $w(X)$, and condition a is replaced by the condition that $w(B_{i_k})$ is maximal among all sets B_j such that $A \uplus B_j$ is independent. Indeed, given σ we can simply use $w(B_j) = j$, and conversely any input where all the weights are distinct enforces a corresponding total ordering on the elements[1]. However, the difference in focus between weights and an ordering is significant, as a total element ordering can carry semantic meaning that is obscured when implemented using weights.

Applying Theorem 1 to vertex cut sparsifiers, we obtain the main result of this paper.

▶ **Theorem 2.** *Given a directed acyclic graph $G = (V, E)$ with terminal sets S, T of size k, we can find a vertex cut sparsifier of G of size $\Theta(k^2)$ algorithmically in time $\tilde{O}((m+n)k^{O(1)})$.*

Our proof initially follows that of Kratsch and Wahlström [13]. Like Kratsch and Wahlström, we compute a "cut-covering set", i.e., a set $Z \subseteq V(G)$ such that for every $A \subseteq S$ and $B \subseteq T$ there is an (A, B)-min cut C with $C \subseteq Z$. However, Kratsch and Wahlström's result is based around *essential vertices*, i.e., vertices $v \in V(G)$ that must be included in any

[1] In the first version of this paper, we presented a proof of Theorem 1 that runs in polynomial time. It was later pointed out to us that this theorem is equivalent to Theorem 3.7 in [6], which runs in near-linear time. We thereby refer the readers to the proof in [6], as our proof shares similar underlying ideas with theirs.

cut-covering set. Using the unordered version of Theorem 1 (see Lemma 1.1 of [11]), they compute a set X of $O(k^3)$ vertices which is guaranteed to include all essential vertices in G. They then eliminate one vertex not in X, recompute the representative family, and repeat until exhaustion. This gives a superlinear running time and a final bound of $|Z| = O(k^3)$. This iterative process for computing Z was required because the initial set X could not be guaranteed to contain *all* vertices of any (A, B)-min cut C, unless that min-cut happens to be unique (i.e., consist only of essential vertices).

We use Theorem 1 to improve over this in two ways. By using the additional power of an ordering, we reduce the bound from $|Z| = O(k^3)$ to $|Z| = O(k^2)$ when G is a DAG. Furthermore, instead of covering some arbitrary min-cut for every pair (A, B), we observe that our construction guarantees that Z contains the unique *closest* (A, B)-min cut to A, i.e., that mincut C for which the set of vertices reachable from A is minimized. This is an interesting consequence that was not previously known, but additionally, it allows us to significantly improve the running time required to compute a cut-covering set. Since the output of Theorem 1 contains all vertices of the closest (A, B)-min cut C for every (A, B), we can compute a cut-covering set Z with a single application of Theorem 1, eliminating the iterative nature of [13] and improving the running time of the procedure.

Finally, to achieve a linear running time, one more obstacle must be overcome. In order to apply Theorem 1, we need to compute a representation for the matroids underlying the result, known as *gammoids*, in time linear in $n + m$. The usual method for representing gammoids goes via the class of *transversal matroids*, however, this requires taking the inverse of an $n \times n$ matrix. Luckily, we observe that an older construction of Mason [17] can be used to represent gammoids more efficiently over DAGs; see Lemma 10. This allows us to compute Z in time linear in $n + m$, and computing the final sparsifier H is then a simple task.

We also show that there are graphs G with k terminals such that any cut sparsifier H requires $\Omega(k^2)$ vertices, both when H is a directed vertex cut sparsifier for a DAG G, and when H is an undirected edge cut sparsifier for an undirected graph G.

Related work. Several different notions of cut sparsifiers have been considered, as well as vertex sparsifiers for other problems. Vertex cut sparsifiers were first introduced by Moitra [18] in the setting of approximation algorithms; see also [15, 2, 4]. Recently, they have found applications in fast graph algorithms, especially in the dynamic setting [5, 1, 3]. Compared to our setting, many of these results replicate min-cuts only approximately, rather than exactly, and most apply only to undirected edge cuts. On the other hand, in the general case (e.g., when working with edge cuts or when terminals are not deletable), there is an important distinction between parameterizing by the *number of terminals* and the *total terminal capacity*. Our setting, with deletable terminals, corresponds to parameterizing edge cuts by the total capacity of the terminal set. Previous results for this setting include Kratsch and Wahlström [13] discussed above and Chuzhoy [4], as well as recent results on terminal multicut sparsification [20]. By contrast, parameterizing by the number of terminals in a setting where terminals have unbounded capacity makes for a much harder sparsification task, and this is the setting that has been most commonly considered in approximation algorithms. Indeed, it is known that any exact cut sparsifier for k terminals with unbounded capacity needs to have at least exponential size in k, and possibly double-exponential [14].

2 Preliminaries

Throughout the paper, all graphs are directed and unweighted. We begin with standard terminology on cuts and cut sparsifiers.

▶ **Definition 3** (Vertex Cut). *Given a directed unweighted graph $G = (V, E)$ with sets $X, Y \subseteq V$, a set $C \subseteq V$ is a* vertex cut *of (X, Y) if after removing C from G, there does not exist a path from a vertex in X to a vertex in Y. We denote the size of a minimum vertex cut between X, Y in G as $\mathrm{mincut}_G(X, Y)$.*

▶ **Definition 4** (Vertex Cut Sparsifier). *Consider a directed unweighted graph $G = (V, E)$ with sets $S, T \subseteq V$. A directed unweighted graph $H = (V', F)$ is a* vertex cut sparsifier *of G if*
(a) *$S, T \subseteq V'$.*
(b) *For all $X \subseteq S, Y \subseteq T$, $\mathrm{mincut}_G(X, Y) = \mathrm{mincut}_H(X, Y)$.*

The problem we consider in this paper is the minimum size of a vertex sparsifier.

▶ **Problem 5** (Minimum Vertex Cut Sparsifier). *Given a directed unweighted graph $G = (V, E)$ with terminal sets $S, T \subseteq V$, what is the minimum number of vertices in a vertex cut sparsifier of G?*

Kratsch and Wahlström [13] obtained a bound for this problem of $O(k^3)$ vertices, where $|S| = |T| = k$. Their application was in the fixed-parameter tractability setting, specifically in constructing kernels for cut-based problems. Our proof utilizes similar matroid-theoretic techniques as in their work. We introduce these techniques next.

▶ **Definition 6** (Matroid). *Given a finite ground set E, a set system $M = (E, I)$ where $I \subseteq \mathcal{P}(E)$ is called a* matroid *if*
(a) *$\varnothing \in I$.*
(b) *For $X, Y \subseteq E$, if $Y \in I$ and $X \subseteq Y$, then $X \in I$.*
(c) *If $X, Y \in I$ and $|X| < |Y|$, then there exists $y \in Y \setminus X$ such that $X \cup \{y\} \in I$.*

Central to our proof is the use of gammoids and their representations.

▶ **Definition 7** (Gammoid). *Given a graph $G = (V, E)$ and a subset of vertices S (which we refer to as the "source vertices"), the* gammoid *on S is the matroid $M = (V, I)$ where I contains all subsets $T \subseteq V$ such that there exist $|T|$ vertex-disjoint paths from S to T in G.*

▶ **Definition 8** (Matroid disjoint union). *Given two matroids on disjoint ground sets, their* matroid disjoint union *is the matroid whose ground set is the union of their ground sets, and a set is independent if the corresponding part in each matroid is independent.*

▶ **Definition 9** (Representable matroid). *Given a field F, a matroid $M = (E, I)$ is* representable *over F if there exists a matrix A over the field F and a bijective mapping from E to the columns of A, such that a set $S \subseteq E$ is independent if and only if its corresponding set of columns of A are linearly independent.*

In particular, it is well known in matroid theory that gammoids are representable in randomized polynomial time; see Marx [16]. However, to control the running time, we revisit an older representation by Mason [17], and note that it leads to a representation in near-linear time in $|V| + |E|$ on DAGs. (Proofs of results marked ★ are deferred to the full version of this paper.)

▶ **Lemma 10** (★ Construction of Gammoid Representation on DAGs). *Given a directed acyclic graph $G = (V, E)$, a set $S \subseteq V$, and $\varepsilon > 0$, with $|V| = n$, $|E| = m$ and $|S| = k$, a representation of the gammoid on S of dimension k over a finite field with entries of bit length $\ell = O(k \log n + \log(1/\varepsilon))$ can be constructed in randomized time $\tilde{O}((n + m)k\ell)$ with one-sided error at most ε, where $\tilde{O}$ hides factors logarithmic in ℓ.*

We also note that the disjoint union of two representable matroids is representable. Given these definitions, we now present the main arguments of this paper.

3 Vertex Cut Sparsifier for DAGs

In this section, we prove our main result, Theorem 2. We first borrow the following key concepts from Kratsch and Wahlström [13].

▶ **Definition 11** (Essential Vertex). *A vertex $v \in V \setminus (S \cup T)$ is called* essential *if there exist $X \subseteq S, Y \subseteq T$ such that v belongs to every minimum vertex cut between X, Y.*

▶ **Definition 12** (Neighborhood Closure). *For a digraph $G = (V, E)$ and a vertex $v \in G$, the neighborhood closure operation is defined by removing v from G and adding an edge from every in-neighbor of v to every out-neighbor of v. The new graph is denoted by $\mathrm{cl}_v(G)$.*

▶ **Definition 13** (Closest Set). *For sets of vertices $X, A \subseteq V$, A is closest to X if A is the unique min-cut between X and A.*

We introduce the following definitions to simplify our discussions.

▶ **Definition 14.** *For sets $X \subseteq S, Y \subseteq T$, let C be a vertex cut for X, Y. Let G' be the subgraph formed by the union of all paths from X to Y. The* left-hand side *of C, denoted $L(C)$, is the set of vertices that are still reachable from X in G' after C is removed. Similarly, the* right-hand side *of C, denoted $R(C)$, is the set of vertices that can still reach Y in G' after C is removed.*

▶ **Definition 15** (Saturation). *Let C be a vertex cut for $X \subseteq S, Y \subseteq T$. For a vertex $v \in C$, we say that (C, v) is saturated by X if there exist $|C| + 1$ paths from X to C that are vertex disjoint except for two paths that both ends at v. Similarly, (C, v) is saturated by Y if there exist $|C| + 1$ paths from C to Y that are vertex disjoint except for two paths that both start at v.*

The following three lemmas follow from [13, Section 5.1]. Their proofs, as presented in [13] and Chapter 11.6 of [7], are included in the Appendix for completeness.

▶ **Lemma 16** (Essential Vertex Lemma). *Let v be an essential vertex with respect to $X \subseteq S$, $Y \subseteq T$. Let C be any minimum vertex cut between X, Y. Then (C, v) is saturated by both X and Y.*

▶ **Lemma 17** (Closure Lemma). *If $v \in V \setminus (S \cup T)$ is not an essential vertex, then $\mathrm{cl}_v(G)$ is a vertex cut sparsifier of G.*

▶ **Lemma 18** (Closest Cut Lemma). *Let C be a vertex cut for $X \subseteq S, Y \subseteq T$ that is closest to X (resp. Y), then for all vertices $v \in C$, (C, v) is saturated by X (resp. Y).*

Using the saturation properties of essential vertices (as in Lemma 16), Kratsch and Wahlström presented a construction of matroids which, combined with the unordered version of Theorem 1, computes a set of vertices P of size $O(k^3)$ that contains all the essential vertices.

They then apply Lemma 17 iteratively to any one vertex not in P and repeat the process. The repetition is required since Lemma 17 may change the essential vertices of the graph.

In our proof, instead of marking only essential vertices we consider all vertices on closest min-cuts. These have weaker saturation properties than essential vertices (as in Lemma 18), but these weaker properties suffice thanks to the ordering imposed in Theorem 1. This ordering, when applied to the topological ordering of a DAG, allows us to mark all vertices of closest min-cuts in a single application of Theorem 1, which eliminates the iterative nature of the previous result. This intuition will be made clear in the following discussion.

We now present our construction, which results in Theorem 2.

▶ **Theorem 19.** *For a directed acyclic graph $G = (V, E)$ with terminal sets S and T, let $k = |S \cup T|$. Then there exists a set of vertices P of size $O(k^2)$ such that for each pair of $X \subseteq S, Y \subseteq T$, the min-$(X, Y)$ cut that is closest to Y is contained in P. This set can be found in time $\tilde{O}(nk^{2\omega-1} + mk^2)$, and a sparsifier on P can then be constructed in the same asymptotic running time.*

Proof. Let $G_R = (V, E_R)$ be the graph G with the direction of all edges reversed. We make the following modification to our graphs G, G_R. For each vertex $v \in V \setminus (S \cup T)$, add a vertex v' into V and for each directed edge $(u, v) \in E$, add (u, v') into E. We refer to v' as the sink-only copy of v. Denote this new directed graph $G' = (V', E')$, we add sink-only copies of vertices to G_R and obtain $G'_R = (V', E'_R)$. Note that G', G'_R are both acyclic. Enumerate V in a reverse topological ordering, namely $V = (v_1, v_2, \cdots, v_n)$ where v_i cannot reach any v_j for $j > i$.

Let $M_1 = (E_1, I_1)$ be the gammoid constructed on the graph G and the set of terminals S in G, and let $M_2 = (E_2, I_2)$ be the gammoid constructed on the graph G'_R and the set of terminals T in G'_R. To distinguish between vertices of E_1 and E_2, we label vertices in E_1 as $v_1, \ldots, v_n$, and elements in E_2 as $\overline{v_1}, \ldots, \overline{v_n}, \overline{v_1}', \ldots, \overline{v_n}'$. Note that E_1 does not contain sink-only copies. For any set of vertices $U \subseteq V$, denote the respective sets in E_1 and E_2 as U_1 and U_2. Let M be the disjoint union of matroids M_1 and M_2, so M is representable and it has rank $O(k)$.

Observe that for any $X \subseteq S$, $Y \subseteq T$, the min-cuts between X and Y in G are the same as in G' because the vertex copies v' we added to G have no outgoing edges. Therefore, G and G' have the same set of closest cuts. We now consider the following constructions. For a min-cut C between X, Y that is closest to Y, and a vertex $v \in C$, define

$$A_{(C,v)} = [(S_1 \setminus X_1) \cup (C_1 \setminus \{v\})] \cup [(T_2 \setminus Y_2) \cup C_2].$$

Let $\tilde{\mathcal{A}}$ consist of $A_{(C,v)}$ for all such cut-vertex pairs. Define

$$\mathcal{F} = \{B_v = \{v, \overline{v}'\} \mid v \in V\}.$$

We pause for a moment to explain the ideas behind these definitions. First of all, note that the algorithm in Theorem 1 only takes as input a family $\mathcal{F}$. Intuitively, one should think of the sets in $\mathcal{F}$ as answers to potential queries, and the sets in $\mathcal{A}$ (defined in Theorem 1) as all queries answered by $\mathcal{F}$. As we will show, the family of queries $\tilde{\mathcal{A}}$ we defined is a subset of $\mathcal{A}$. If we can further show that for each v in a closest min-cut C, B_v is the unique answer that the algorithm will find for query $A_{(C,v)}$, then we know that the output of the algorithm must contain all vertices of all closest min-cuts, because all queries in $\mathcal{A} \supseteq \tilde{\mathcal{A}}$ must be answered.

More specifically, for each query $A_{(C,v)}$, Theorem 1 promises to find the answer B_u to $A_{(C,v)}$ (which means $A_{(C,v)} \uplus B_u$ is independent in M) that is the last answer according to the reverse topological ordering on $\mathcal{F}$. This means for all $w > u$, $A_{(C,v)} \uplus B_w$ is dependent in M. Since our goal is for Theorem 1 to output a set containing all vertices on all closest min-cuts, we want to show that B_v is the last answer in the ordering to $A_{(C,v)}$. This is precisely captured in the following claim.

▷ Claim 20. For each $A = A_{(C,v)}$, B_v is the unique set in $\mathcal{F}$ such that
- $A \uplus B_v$ is independent in M, and
- for all $u > v$ in the reverse topological ordering of V, $A \uplus B_u$ is dependent in M.

Proof. We first show that $A \uplus B_v$ is independent. Since M is a disjoint union matroid, we need to show that $(A \cap E_1) \uplus \{v\} = (S_1 \setminus X_1) \cup C_1$ is independent in M_1 and $(A \cap E_2) \uplus \{\overline{v}'\} = (T_2 \setminus Y_2) \cup C_2 \cup \{\overline{v}'\}$ is independent in M_2.

Since both M_1 and M_2 are gammoids, we need to show existence of vertex disjoint paths from S_1 to $(S_1 \setminus X_1) \cup C_1$. First note that singleton paths can cover all vertices in $S_1 \setminus X_1$. Since C_1 is a min-cut between X_1 and Y_1, by duality there exist vertex disjoint paths from X_1 to C_1. Therefore $(S_1 \setminus X_1) \cup C_1$ is independent in M_1. Similarly, singleton paths can cover all vertices in $T_2 \setminus Y_2$. It suffices for us to show the existence of vertex disjoint paths from Y_2 to $C_2 \cup \{\overline{v}'\}$.

By Lemma 18, there exist $|C_2| + 1$ paths from Y_2 to C_2 that are vertex disjoint except for two paths that both ends at $\overline{v}$. Therefore, we can redirect one of these two paths to end at $\overline{v}'$, and we obtain $|C_2| + 1$ vertex disjoint paths from Y_2 to $C_2 \cup \{\overline{v}'\}$. This proves independence.

Now fix $u > v$ in the reverse topological ordering, so that there does not exist a path from v to u. We want to show that either $(A \cap E_1) \uplus \{u\}$ is dependent in M_1, or $(A \cap E_2) \uplus \{\overline{u}'\}$ is dependent in M_2. Consider four possible cases:
- u is not on any path from X to Y. Assume for the sake of contradiction that both $(A \cap E_1) \uplus \{u\}$ and $(A \cap E_2) \uplus \{\overline{u}'\}$ are independent. Then there must exist a path from X to u and a path from u to Y, which forms a path from X to Y through u (since G is acyclic), contradiction.
- $u \in L(C)$ (see Definition 14). Then any path from u to Y (or from Y to u in G_R) must intersect with C, which means there does not exist vertex disjoint paths from Y to $C \cup \{u\}$ in G_R. Therefore $(A \cap E_2) \uplus \{\overline{u}'\}$ is dependent.
- $u \in R(C)$. Then any path from X to u must intersect C. Assume for the sake of contradiction that $(A \cap E_1) \uplus \{u\}$ is independent, then there exist vertex disjoint paths from X to $C \setminus \{v\} \cup \{u\}$, which means there is a path from X to u that goes through v. However, since $u > v$ in the topological ordering, there does not exist paths from v to u. This is a contradiction, so $(A \cap E_1) \uplus \{u\}$ is dependent.
- $u \in C$. Then $u \in (A \cap E_1)$, which implies $(A \cap E_1) \uplus \{u\}$ contains two copies of u. Therefore it is dependent.

This completes the proof. ◁

We now apply Theorem 1 on $\mathcal{F}$ with a reverse topological ordering, and we note that the family $\tilde{\mathcal{A}}$ we defined in this proof is a subfamily of $\mathcal{A}$ defined in Theorem 1. Let P be the collection of vertices that Theorem 1 finds. Then by the above claim, for each pair of $X \subseteq S, Y \subseteq T$ and their min-cut C closest to Y, all vertices in C must be in P. Note that this also implies that all essential vertices are in P.

To construct the final sparsifier H on P, for each vertex $u \in P$, we run a depth-first search starting at u on the graph G_u, defined to be G minus the out-edges of vertices in $P \setminus \{u\}$. For each vertex $v \in P \setminus \{u\}$ that is reachable from u in G_u, we add an edge (u, v) to H. Note that this procedure returns the same graph H as the one that sequentially applies Lemma 17 on all vertices not in P, but achieves a shorter runtime of $O(k^2 m)$. To see the equivalence, observe that in both cases, there is an edge (u, v) in the final sparsifier if and only if there is a path from u to v in G whose internal vertices are disjoint from P. We conclude that the output graph H is a valid sparsifier.

For the final running time, note that computing the gammoids takes time $\tilde{O}((m+n)k^2)$ by Lemma 10 with $\varepsilon = 1/n^{O(1)}$, and computing the representative sets takes time $\tilde{O}(nk^{2\omega-1})$ by taking $d = k$ and $s = 2$ in Theorem 1. ◀

We make a few remarks regarding this proof. Intuitively, the gammoid M_2 and its respective query $A_{(C,v)} \cap E_2 = (T_2 \setminus Y_2) \cup C_2$ is used to filter out all vertices on the left-hand side of C, because no vertex in $L(C)$ can reach T without crossing C. The gammoid M_1 and its query $A_{(C,v)} \cap E_1 = (S_1 \setminus X_1) \cup (C_1 \setminus \{v\})$, however, is different, because the query itself allows terminals in X to reach the right-hand side of C through v. If we do not impose a reverse topological ordering on the vertices and use the unordered version of Theorem 1 (Lemma 1.1 in [11]), our proof will fail – there may exists $u \in R(C)$ reachable from v such that $A_{(C,v)} \uplus B_u$ is independent, and the algorithm may not find B_v as the answer. By imposing the reverse topological ordering, we demand the algorithm to find the answer last in the ordering, thereby ensuring the algorithm discovering B_v.

This is the critical difference between our proof and Kratsch and Wahlström's proof. In Kratsch and Wahlström's paper, they presented a construction with three matroids – one gammoid on G' and S_1 (note that our first gammoid is constructed on G and S_1), one gammoid on G'_R and T_2, and one uniform matroid of rank k on V. They then utilized the property that essential vertices can be saturated from both sides (see Lemma 16) and constructed queries similar to our definition of $A_{(C,v)}$. Their first (resp. second) gammoid serves to filter out $R(C)$ (resp. $L(C)$), and they use the uniform matroid to filter out vertices $u \neq v \in C$. We managed to merge their first gammoid and uniform matroid with an asymmetrical construction, thereby proving a stronger bound.

We finally remark that in our proof, we never explicitly compute the queries $A_{(C,v)}$. As mentioned previously, the algorithm in Theorem 1 only takes as input the family $\mathcal{F}$, and we are simply showing that given our construction of $\mathcal{F}$, $\tilde{\mathcal{A}}$ is a subset of queries answered. If one can show that another meaningful set of queries are answered by the same $\mathcal{F}$, then they can derive more properties of the output set of the algorithm (while the output set itself remains unchanged).

We now present tight lower bound constructions in the following section.

4 Lower Bound Constructions

In this section we present two constructions graphs for which any vertex cut sparsifier requires $\Omega(k^2)$ vertices. The first construction is presented by Kratsch and Wahlström [13] (construction found in arXiv preprint only); note that the graph is a DAG.

▶ **Lemma 21** (★). *Let S and T be two vertex sets of size $2k$. Enumerate them as*

$$S = \{v_1, v_1', v_2, v_2', \cdots, v_k, v_k'\}, \qquad T = \{u_1, u_1', u_2, u_2', \cdots, u_k, u_k'\}.$$

For each $i, j \in [k]$, create a vertex $w_{i,j}$ and add edges from v_i, v_i' to $w_{i,j}$, and from $w_{i,j}$ to u_j, u_j'. Any vertex cut sparsifier for the resulting graph requires $\Omega(k^2)$ vertices.

Next, we show the same bound holds also for *edge cut* sparsifiers, when both G and its sparsifier H are undirected. For simplicity, we also restrict ourselves to just one terminal set T.

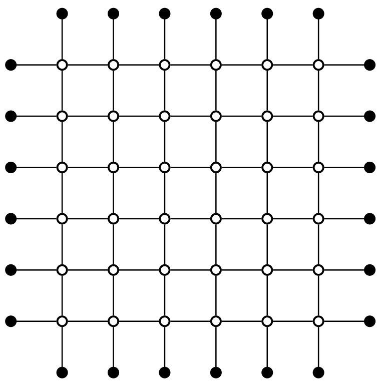


Figure 1 The lower bound construction for $k = 6$.

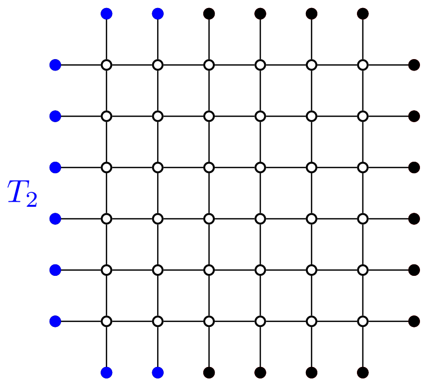


Figure 2 The set T_i for $i = 2$.

▶ **Definition 22** ((Undirected) Edge Cut Sparsifier). *Consider an unweighted, undirected graph $G = (V, E)$ with terminal set $T \subseteq V$. An unweighted, undirected graph $H = (V', F)$ is an* edge cut sparsifier *of G if*
(a) $T \subseteq V'$.
(b) *For all disjoint $X, Y \subseteq T$,* $\text{emincut}_G(X, Y) = \text{emincut}_H(X, Y)$.

Consider the k-by-k grid with leaf terminals attached in Figure 1: begin with a k-by-k grid of non-terminals, and add the following leaf (degree-1) terminals:
1. one leaf terminal adjacent to each vertex on the top row, called the *top* terminals,
2. one leaf terminal adjacent to each vertex on the bottom row, called the *bottom* terminals,
3. one leaf terminal adjacent to each vertex on the left column, called the *left* terminals, and
4. one leaf terminal adjacent to each vertex on the right column, called the *right* terminals.

Note that non-terminals on the corner of the grid have two terminals attached. This concludes the construction of the terminal set T. We now show that any edge cut sparsifier of G, even with directed edges allowed, has at least k^2 vertices. Since G has $O(k)$ terminals, this proves the quadratic lower bound.

▶ **Lemma 23.** *Any (undirected) edge cut sparsifier of G has $\Omega(k^2)$ vertices.*

For the rest of this section, we prove Lemma 23. For an undirected graph G and a subset of vertices S, we define $\partial_G S$ as the set of edges with exactly one endpoint in S.

Consider a directed sparsifier H with vertex set $V' \supseteq T$. For $0 \leq i \leq k$, let $T_i \subseteq T$ be following set of terminals: all of the left terminals, plus the first i top and bottom terminals, counting from the left (see Figure 2). Let $C_i \subseteq V'$ be T_i's side of the mincut between $T_i, T \setminus T_i$

in the sparsifier H, which must have $|\partial C_i| = |\partial(V' \setminus C_i)| = k$, since $\operatorname{emincut}_G(T_i, T \setminus T_i)$ is k and H is a sparsifier of G. Similarly, let $T'_i \subseteq T$ be all the top terminals, plus the first i left and right terminals, counting from the top, and let $R_i \subseteq V'$ be T'_i's side of the mincut between $T'_i, T \setminus T'_i$ in H, so that $|\partial R_i| = |\partial(V' \setminus R_i)| = k$.

▷ Claim 24. Without loss of generality, we may assume that $C_0 \subseteq C_1 \subseteq \cdots \subseteq C_k$ and $R_0 \subseteq R_1 \subseteq \cdots \subseteq R_k$.

Proof. We only prove the statement for C_i; the one for R_i follows by symmetry of G. Suppose for contradiction that $C_i \setminus C_{i+1} \neq \emptyset$. By submodularity,

$$|\partial C_i| + |\partial C_{i+1}| \geq |\partial(C_i \cap C_{i+1})| + |\partial(C_i \cup C_{i+1})|.$$

Since $C_i \cap C_{i+1}$ is a $(T_i, T \setminus T_i)$-cut in H and $C_i \cup C_{i+1}$ is a $(T_{i+1}, T \setminus T_{i+1})$-cut in H, their cut values $\partial(C_i \cap C_{i+1})$ and $\partial(C_i \cup C_{i+1})$ are at least k. Therefore,

$$k + k = |\partial C_i| + |\partial C_{i+1}| \geq |\partial(C_i \cap C_{i+1})| + |\partial(C_i \cup C_{i+1})| \geq k + k,$$

so the inequality must be an equality. It follows that we can replace C_i with $C_i \cap C_{i+1}$, which is still a $(T_i, T \setminus T_{i+1})$-cut in H, and we can also replace C_{i+1} with $C_i \cup C_{i+1}$. While there exists an i such that $C_i \setminus C_{i+1} \neq \emptyset$, we perform the replacement; this can only happen a finite number of times, since the quantity $\sum_{i=0}^{k} |C_i|^2$ increases by at least 1 each time and has an upper limit. ◁

For each $1 \leq i, j < k$, define $S_{i,j} \subseteq V'$ as $S_{i,j} = (C_{i+1} \setminus C_i) \cap (R_{j+1} \setminus R_j)$; see Figure 3a. Our goal is to prove that $S_{i,j} \neq \emptyset$ for all i, j; since the sets are disjoint over all $1 \leq i, j \leq k$, this implies the k^2 bound.

▷ Claim 25. There are no edges between $S_{i,j}$ and $S_{i',j'}$ for $i \neq i'$ and $j \neq j'$.

Proof. First, consider some $1 \leq i, i', j, j' \leq k$ where $i < i'$ and $j < j'$ (see Figure 3a). By submodularity on the sets C_{i+1} and $R_{j'+1}$,

$$|\partial C_{i+1}| + |\partial R_{j'+1}| \geq |\partial(C_{i+1} \cap R_{j'+1})| + |\partial(C_{i+1} \cup R_{j'+1})|.$$

Since $\partial(C_{i+1} \cap R_{j'+1})$ is a $(T_{i+1} \cap T'_{j'+1}, T \setminus (T_{i+1} \cap T'_{j'+1}))$-cut, its value is at least

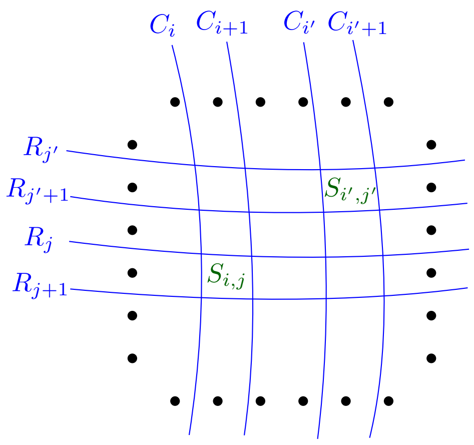
$$\operatorname{emincut}_G(T_{i+1} \cap T'_{j'+1}, T \setminus (T_{i+1} \cap T'_{j'+1})) = (i+1) + (j'+1).$$

Similarly, $|\partial(C_{i+1} \cup R_{j'+1})| \geq k - (i+1) + k - (j'+1)$. It follows that

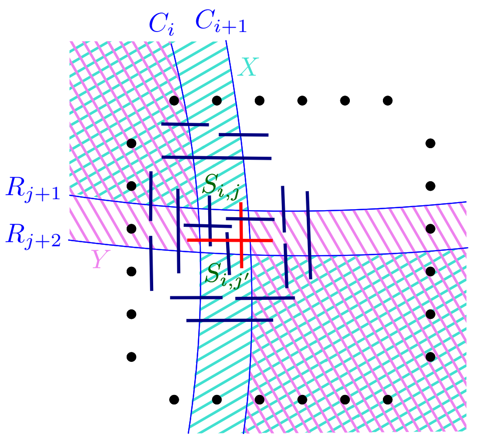
$$\begin{aligned} k + k = |\partial C_{i+1}| + |\partial R_{j'+1}| &\geq |\partial(C_{i+1} \cap R_{j'+1})| + |\partial(C_{i+1} \cup R_{j'+1})| \\ &\geq (i+1) + (j'+1) + k - (i+1) + k - (j'+1) \\ &= k + k, \end{aligned}$$

so the inequality must be an equality. It follows that there are no edges between $C_{i+1} \setminus R_{j'+1}$ and $R_{j'+1} \setminus C_{i+1}$, since those edges would make the submodularity inequality strict. In particular, there are no edges between $S_{i,j}$ and $S_{i',j'}$.

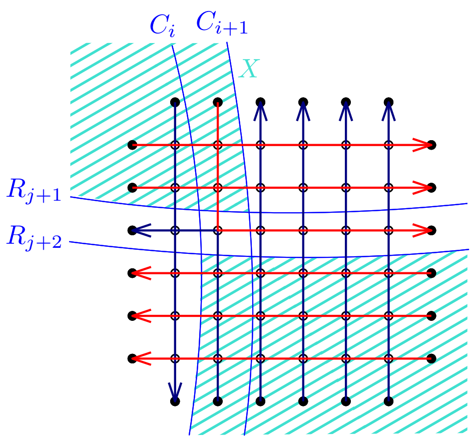
Finally, the $i < i'$ and $j < j'$ assumptions can be removed essentially by symmetry, replacing C_{i+1} by $V' \setminus C_{i+1}$ or $R_{j'+1}$ by $V' \setminus R_{j'+1}$ (or both). ◁



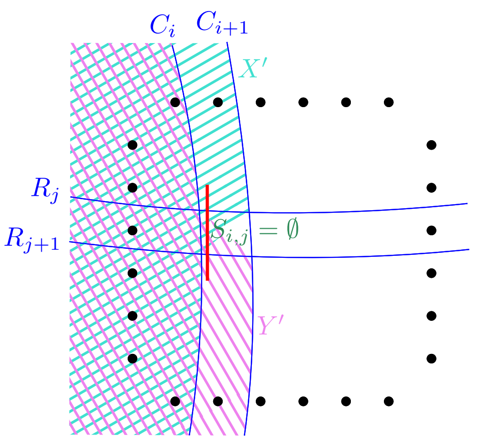
(a) The setting for the proof of Claim 25.



(b) The definitions of X and Y, and the different types of edges that are allowed by Claim 25 and leave or enter the sets R_{j+1}, R_{j+2}, C_i, C_{i+1}. The long edges marked in red make the inequality $|\partial_H X| + |\partial_H Y| \le |\partial_H R_{j+1}| + |\partial_H R_{j+2}| + |\partial_H C_i| + |\partial_H C_{i+1}|$ strict.



(c) The cut and flow that establishes that $\text{emincut}_G(X \cap T, T \setminus X) = 2k$. The flow is drawn in red and dark blue.



(d) The definitions of X' and Y'. The red edge makes the submodularity inequality strict.

Figure 3 Figures for Lemma 23.

▷ Claim 26. There are no edges between $S_{i,j}$ and $S_{i,j'}$ for $|j - j'| \ge 2$. Similarly, there are no edges between $S_{i,j}$ and $S_{i',j}$ for $|i - i'| \ge 2$.

Proof. We first prove the inequality for $S_{i,j}$ and $S_{i,j'}$ for $j' \ge j + 2$. As shown in Figure 3b, define $X = (R_{j+1} \cap C_{i+1}) \cup ((V \setminus R_{j+2}) \cap (V \setminus C_i))$ and $Y = (R_{j+2} \cap C_i) \cup ((V' \setminus R_{j+1}) \cap (V' \setminus C_{i+1}))$. We now claim the inequality

$$|\partial_H X| + |\partial_H Y| \le |\partial_H R_{j+1}| + |\partial_H R_{j+2}| + |\partial_H C_i| + |\partial_H C_{i+1}| \tag{1}$$

by examining each type of edge in Figure 3b and its contribution to both sides of the inequality:

1. Each edge between $C_i \cap R_{j+1}$ and $(C_{i+1} \setminus C_i) \cap R_{j+1}$ contributes 1 to $|\partial_H Y|$ and 1 to $|\partial_H C_i|$.
2. Each edge between $C_{i+1} \cap R_{j+1}$ and $(V \setminus C_{i+1}) \cap R_{j+1}$ contributes 1 to $|\partial_H X|$ and 1 to $|\partial_H C_{i+1}|$.

3. Each edge between $C_i \cap R_{j+1}$ and $(V \setminus C_{i+1}) \cap R_{j+1}$ contributes 1 to $|\partial_H X|$ and $|\partial_H Y|$ and 1 to $|\partial_H C_i|$ and $|\partial_H C_{i+1}|$.
4. Each edge between $C_i \cap R_{j+1}$ and $C_i \cap (R_{j+2} \setminus R_{j+1})$ contributes 1 to $|\partial_H X|$ and 1 to $|\partial_H R_{j+1}|$.
5. Each edge between $C_i \cap R_{j+2}$ and $C_i \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H Y|$ and 1 to $|\partial_H R_{j+1}|$.
6. Each edge between $C_i \cap R_{j+1}$ and $C_i \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H X|$ and $|\partial_H Y|$ and 1 to $|\partial_H R_{j+1}|$ and $|\partial_H R_{j+2}|$.
7. Each edge between $C_i \cap (V \setminus R_{j+2})$ and $(C_{i+1} \setminus C_i) \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H X|$ and 1 to $|\partial_H C_i|$.
8. Each edge between $C_{i+1} \cap (V \setminus R_{j+2})$ and $(V \setminus C_{i+1}) \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H Y|$ and 1 to $|\partial_H C_{i+1}|$.
9. Each edge between $C_i \cap (V \setminus R_{j+2})$ and $(V \setminus C_{i+1}) \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H X|$ and $|\partial_H Y|$ and 1 to $|\partial_H C_i|$ and $|\partial_H C_{i+1}|$.
10. Each edge between $(V \setminus C_{i+1}) \cap R_{j+1}$ and $(V \setminus C_{i+1}) \cap (R_{j+2} \setminus R_{j+1})$ contributes 1 to $|\partial_H Y|$ and 1 to $|\partial_H R_{j+1}|$.
11. Each edge between $(V \setminus C_{i+1}) \cap R_{j+2}$ and $(V \setminus C_{i+1}) \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H X|$ and 1 to $|\partial_H R_{j+1}|$.
12. Each edge between $(V \setminus C_{i+1}) \cap R_{j+1}$ and $(V \setminus C_{i+1}) \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H X|$ and $|\partial_H Y|$ and 1 to $|\partial_H R_{j+1}|$ and $|\partial_H R_{j+2}|$.
13. Each edge between $C_i \cap (R_{j+2} \setminus R_{j+1})$ and $(C_{i+1} \setminus C_i) \cap (R_{j+2} \setminus R_{j+1})$ contributes 1 to $|\partial_H Y|$ and 1 to $|\partial_H C_i|$.
14. Each edge between $(C_{i+1} \setminus C_i) \cap (R_{j+2} \setminus R_{j+1})$ and $(V \setminus C_{i+1}) \cap (R_{j+2} \setminus R_{j+1})$ contributes 1 to $|\partial_H Y|$ and 1 to $|\partial_H C_{i+1}|$.
15. Each edge between $C_i \cap (R_{j+2} \setminus R_{j+1})$ and $(V \setminus C_{i+1}) \cap (R_{j+2} \setminus R_{j+1})$ contributes 1 to $|\partial_H C_i|$ and 1 to $|\partial_H C_{i+1}|$.
16. Each edge between $(C_{i+1} \setminus C_i) \cap R_{j+1}$ and $(C_{i+1} \setminus C_i) \cap (R_{j+2} \setminus R_{j+1})$ contributes 1 to $|\partial_H X|$ and 1 to $|\partial_H R_{j+1}|$.
17. Each edge between $(C_{i+1} \setminus C_i) \cap (R_{j+2} \setminus R_{j+1})$ and $(C_{i+1} \setminus C_i) \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H X|$ and 1 to $|\partial_H R_{j+2}|$.
18. Each edge between $(C_{i+1} \setminus C_i) \cap R_{j+1}$ and $(C_{i+1} \setminus C_i) \cap (V \setminus R_{j+2})$ contributes 1 to $|\partial_H R_{j+1}|$ and 1 to $|\partial_H R_{j+2}|$.

All of the above types of edges contribute the same to both sides of (1) except those of type 15 and 18, namely those indicated by long red edges in Figure 3b. We call such edges *red*. Then, the inequality (1) is strict if any only if red edges are present.

Since all edges between $S_{i,j}$ and $S_{i,j'}$ are red, it suffices to show that (1) is actually an equality, which would exclude all red edges and hence all edges between $S_{i,j}$ and $S_{i,j'}$ as well. Observe that $\text{emincut}_G(X \cap T, T \setminus X) = 2k$, as seen in Figure 3c, which shows a cut and a flow both of value $2k$. Similarly, $\text{emincut}_G(Y \cap T, T \setminus Y) = 2k$. Since H is a sparsifier of G, we must have

$$\begin{aligned} 2k + 2k = |\partial_H X| + |\partial_H Y| &\le |\partial_H R_{j+1}| + |\partial_H R_{j+2}| + |\partial_H C_i| + |\partial_H C_{i+1}| \\ &= k + k + k + k. \end{aligned}$$

In other words, the inequality is tight, as desired.

The case for $S_{i,j}$ and $S_{i',j}$ for $i' \ge i + 2$ is similar. Note that edges between $S_{i,j}$ and $S_{i',j}$ correspond to the red horizontal edges in Figure 3b (for different values of i, j), which we have already shown do not exist. $\lhd$

▷ Claim 27. For each $1 \le i, j \le k$, we have $S_{i,j} \neq \emptyset$.

Proof. Suppose for contradiction that $S_{i,j} = \emptyset$ for some $1 \le i, j \le k$. As shown in Figure 3d, define the sets $X' = C_i \cup (C_{i+1} \cap R_j)$ and $Y' = C_i \cup (C_{i+1} \setminus R_{j+1})$. We have $X' \cap Y' = C_i$, and since $S_{i,j} = \emptyset$ by assumption, we also have $X' \cup Y' = C_{i+1}$. By submodularity,

$$|\partial_H X'| + |\partial_H Y'| \ge |\partial_H (X' \cap Y')| + |\partial_H (X' \cup Y')| = |\partial_H C_i| + |\partial_H C_{i+1}|.$$

Moreover, the inequality is tight because the only types of edges that can make the inequality strict (marked red in Figure 3d) are prohibited by Claim 26. Since H is a sparsifier of G,

$$\text{emincut}_G(X' \cap T, T \setminus X') + \text{emincut}_G(Y' \cap T, T \setminus Y') \le |\partial_H X'| + |\partial_H Y'| = |\partial_H C_i| + |\partial_H C_{i+1}| = 2k.$$

However, it is not hard to see that $\text{emincut}_G(X' \cap T, T \setminus X') = \text{emincut}_G(Y' \cap T, T \setminus Y') = k+1$, a contradiction. ◁

5 Conclusions

We showed that every unweighted, directed acyclic graph G with k terminals admits a vertex cut sparsifier H with $O(k^2)$ vertices, assuming that the terminals are deletable. This improves the previous result by Kratsch and Wahlström of $O(k^3)$ vertices, for general directed graphs [13]. Furthermore, the sparsifier can be computed in near-linear time in the size of G, specifically in time $O((m+n)k^{O(1)})$ plus $O((m+n)k^{O(1)})$ field operations over a finite field with entries of bitlength $O(k \log n)$, where $n = |V(G)|$ and $m = |E(G)|$. This improves over previous work [13], whose time complexity was not explicitly given but is at least $O(n^{\omega+1})$ due to the repeated construction of a representation of a gammoid.

Furthermore, we showed that $\Omega(k^2)$ vertices in a sparsifier may be required, both for vertex cuts in DAGs and for the seemingly simpler setting of undirected edge cuts. However, we leave it open whether such a bound applies to the mixed setting, where we want to preserve undirected edge cuts in the input graph G but allow the sparsifier to be a directed graph.

More importantly, we leave open the question of whether vertex cut sparsifiers of $O(k^2)$ vertices exist for general directed graphs. We conjecture that $O(k^2)$ is the correct bound for general directed graphs, but we were not able to find a proof.

References

1 Parinya Chalermsook, Syamantak Das, Yunbum Kook, Bundit Laekhanukit, Yang P Liu, Richard Peng, Mark Sellke, and Daniel Vaz. Vertex sparsification for edge connectivity. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1206–1225. SIAM, 2021.

2 Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 265–274. IEEE, 2010.

3 Li Chen, Gramoz Goranci, Monika Henzinger, Richard Peng, and Thatchaphol Saranurak. Fast dynamic cuts, distances and effective resistances via vertex sparsifiers. In *61st IEEE Annual Symposium on Foundations of Computer Science*, pages 1135–1146. IEEE, 2020.

4 Julia Chuzhoy. On vertex sparsifiers with Steiner nodes. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 673–688, 2012.

5 David Durfee, Yu Gao, Gramoz Goranci, and Richard Peng. Fully dynamic spectral vertex sparsifiers and applications. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 914–925, 2019.

6 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016.

7 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

8 Peter Frankl. An extremal problem for two families of sets. *Eur. J. Comb.*, 3(2):125–127, 1982. doi:10.1016/S0195-6698(82)80025-5.

9 Eva-Maria C Hols and Stefan Kratsch. A randomized polynomial kernel for subset feedback vertex set. *Theory of Computing Systems*, 62(1):63–92, 2018.

10 Stefan Kratsch. A randomized polynomial kernelization for vertex cover with a smaller parameter. *SIAM Journal on Discrete Mathematics*, 32(3):1806–1839, 2018.

11 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 450–459. IEEE, 2012.

12 Stefan Kratsch and Magnus Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms (TALG)*, 10(4):1–15, 2014.

13 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020.

14 Robert Krauthgamer and Inbal Rika. Mimicking networks and succinct representations of terminal cuts. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1789–1799. SIAM, 2013.

15 Konstantin Makarychev and Yury Makarychev. Metric extension operators, vertex sparsifiers and Lipschitz extendability. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 255–264. IEEE, 2010.

16 Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009.

17 John H Mason. On a class of matroids arising from paths in graphs. *Proceedings of the London Mathematical Society*, 3(1):55–74, 1972.

18 Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 3–12. IEEE, 2009.

19 Burkhard Monien. How to find long paths efficiently. In *North-Holland Mathematics Studies*, volume 109, pages 239–254. Elsevier, 1985.

20 Magnus Wahlström. On quasipolynomial multicut-mimicking networks and kernelization of multiway cut problems. In *ICALP*, volume 168 of *LIPIcs*, pages 101:1–101:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.