Tight Conditional Lower Bounds for Vertex Connectivity Problems

Zhiyi Huang[1], Yaowei Long[2], Thatchaphol Saranurak*[2], and Benyu Wang[1]

[1]Tsinghua University
[2]University of Michigan

April, 2023

Abstract

We study the fine-grained complexity of graph connectivity problems in unweighted undirected graphs. Recent development shows that all variants of *edge connectivity* problems, including single-source-single-sink, global, Steiner, single-source, and all-pairs connectivity, are solvable in $m^{1+o(1)}$ time, collapsing the complexity of these problems into the almost-linear-time regime. While, historically, *vertex connectivity* has been much harder, the recent results showed that both single-source-single-sink and global vertex connectivity can be solved in $m^{1+o(1)}$ time, raising the hope of putting all variants of vertex connectivity problems into the almost-linear-time regime too.

We show that this hope is impossible, assuming conjectures on finding 4-cliques. Moreover, we essentially settle the complexity landscape by giving tight bounds for *combinatorial* algorithms in dense graphs. There are three separate regimes:

1. all-pairs and Steiner vertex connectivity have complexity $\hat{\Theta}(n^4)$,
2. single-source vertex connectivity has complexity $\hat{\Theta}(n^3)$, and
3. single-source-single-sink and global vertex connectivity have complexity $\hat{\Theta}(n^2)$.

For graphs with general density, we obtain tight bounds of $\hat{\Theta}(m^2)$, $\hat{\Theta}(m^{1.5})$, $\hat{\Theta}(m)$, respectively, assuming Gomory-Hu trees for element connectivity can be computed in almost-linear time.

*Supported by NSF CAREER grant 2238138.

1 Introduction

Vertex connectivity and edge connectivity are central concepts in graph theory. In an unweighted undirected graph G with n vertices and m edges, the *vertex connectivity* (*edge connectivity*) between two vertices u, v is the maximum number of internally vertex-disjoint (edge-disjoint) paths from u to v. Efficient algorithms for variants of both problems have been extensively studied for at least half a century [FF56, Kle69, GH61, GR98, CKM+11, She13, KLOS14, Mad16, vdBLN+20, CKL+22].

Until very recently, the graph algorithm community has reached a very satisfying conclusion on the complexity of edge connectivity problems: *all variants of edge connectivity problems can be solved in almost-linear time*. There are five variants of the connectivity problems studied in the literature, including (1) global, (2) single-source single-sink, (3) Steiner, (4) single-source, and (5) all-pairs connectivity. See the detailed definitions in Section 2. Among these problems, all-pairs connectivity is the hardest via straightforward reductions. For the global edge connectivity problem, Karger showed in 2000 that the problem admits a near-linear time algorithm [Kar00]. In the recent line of work [LP20, AKT21b, LP21, AKT21a, LPS21, AKL+22], Abboud et al. [AKL+22] finally showed that even the all-pairs edge connectivity case could be reduced to the single-source single-sink case, which solvable in almost-linear time via the recent max flow result by Chen et al. [CKL+22]. This finally puts all five problems on edge connectivity into the almost-linear time regime and raises the following hope:

Can we solve all variants of vertex connectivity problems in almost-linear time too?

Historically, the vertex connectivity problems have been much more difficult than their edge connectivity counterpart. Furthermore, Abboud et al. [AKT20] showed that the all-pairs vertex connectivity in *weighted* graphs with $\tilde{O}(n)$ edges requires $\hat{\Omega}(n^3)$ time assuming SETH.[1] In *directed* unweighted graphs, Abboud et al. [AGI+19] also showed that the problem even requires $\hat{\Omega}(n^4)$ for combinatorial algorithms and $\hat{\Omega}(n^{\omega+1})$ time for general algorithms. Thus, at least in weighted or directed graphs, the problem does not admit almost-linear algorithms.

But, again, the recent development on vertex connectivity in the standard *unweighted undirected* graphs raises the hope for almost-linear time algorithms. Li et al. [LNP+21] showed how to compute global vertex connectivity using polylogarithmic calls to max flows, which implied an $\hat{O}(m)$-time algorithm via the max flow algorithm of [CKL+22] and improved upon the known $\tilde{O}(mn)$ bound by [HRG00]. Indeed, the recent max flow algorithm [CKL+22] also implies a $\hat{O}(m)$-time algorithm for the (s,t)-vertex connectivity problem. Until now, there are still no nontrivial lower bounds for Steiner, single-source, and all-pairs vertex connectivity in unweighted graphs, and the technique of [AKT20] is quite specific for weighted graphs. The complexity landscape for vertex connectivity problems is still very unclear.

1.1 Our Results

We give a firm negative answer to the above open problem. Based on well-known conjectures, we settle the complexity landscape for all five vertex connectivity problems in dense graphs by giving tight bounds for *combinatorial* algorithms, which generally refer to algorithms that do not use fast matrix multiplication. We can obtain tight bounds even in general graphs by assuming a possible hypothesis about computing Gomory-Hu trees for element connectivity. Below, we discuss our results in more detail.

[1] We use $\tilde{O}(\cdot)$ to hide a polylog(n) factor and $\hat{O}(\cdot)$ to hide a $n^{o(1)}$ factor.

All-pairs vertex connectivity (Section 3). Our first result is a conditional lower bound of the all-pair vertex connectivity (APVC) problem based on the *4-clique conjecture*, which postulates that the running time for deciding the existence of a 4-clique in a graph must be at least $\hat{\Omega}(n^4)$ time for combinatorial algorithms [BGL17] and $\hat{\Omega}(n^{\omega(1,2,1)}) = \Omega(n^{3.25})$ time for general algorithms [DW22].[2]

Theorem 1.1. *Assuming the 4-clique conjecture, the all-pairs vertex connectivity problem on an undirected unweighted graph with n vertices requires $\hat{\Omega}(n^4)$ time for combinatorial algorithms, and $\Omega(n^{3.25})$ time for general algorithms.*

Theorem 1.1 gives the first super cubic lower bounds for all-pairs vertex connectivity problems in the standard undirected case. Moreover, the bound is tight for *combinatorial* algorithms. Indeed, a naive algorithm is to call max flow $O(n^2)$ times, one for each pair of vertices, which takes $\hat{O}(mn^2) = \hat{O}(n^4)$ time.

It is interesting to compare APVC with the *all-pairs shortest paths* (APSP) problem, a central problem in fine-grained complexity. It is conjectured that the right complexity of APSP in weighted graphs is $\hat{\Theta}(n^3)$ (see e.g. [WW18]). Theorem 1.1 shows that, for general algorithms, APVC in unweighted graphs is strictly harder than APSP in weighted graphs, assuming $\omega > 2$,

Steiner vertex connectivity (Section 4). Next, we study the *Steiner vertex connectivity* problem. In this problem, given a set of vertices T, we need to compute the minimum vertex connectivity among all pairs of vertices in T. Even though Steiner vertex connectivity looks much easier than APVC, we can extend the same lower bound of Theorem 1.1 for APVC to work for the Steiner case too.

Towards this hardness, we propose a variant of the 4-clique conjecture, the *edge-universal* 4-clique conjecture, which postulates that, given any subset of demand edges $E_{\text{dem}} \subseteq E(G)$, checking if every edge in E_{dem} is contained in a 4-clique requires $\hat{\Omega}(n^4)$ time for combinatorial algorithms (see Conjecture 2.3 for details).[3]

Theorem 1.2. *Assuming the edge-universal 4-clique conjecture, the Steiner vertex connectivity problem on an undirected unweighted graph with n vertices requires $\hat{\Omega}(n^4)$ time for combinatorial algorithms.*

We note that our reduction would imply hardness for general algorithms too if we conjectured the hardness of the edge-universal 4-clique problem for general algorithms.

Upper bounds for general density (Section 5). On sparse graphs, we observe that there is still some discrepancy between our lower bounds and the naive algorithm. More precisely, we observe that our lower bounds for combinatorial algorithms for all-pairs and Steiner vertex connectivity can be easily extended to $\hat{\Omega}(m^2)$ in a graph with m edges. However, the naive algorithm requires $\hat{O}(mn^2)$ time. For example, in sparse graphs, the naive algorithm takes $\hat{O}(n^3)$ time, while our lower bound is $\hat{\Omega}(n^2)$.

We fix the above discrepancy by showing improved algorithms in sparse graphs via the following result.

[2] $\omega(1,2,1)$ is the exponent of multiplying an $n \times n^2$ matrix by an $n^2 \times n$ matrix.

[3] Note that this problem is at least as hard as the problem when $E_{\text{dem}} = E(G)$, i.e. when we need to check if every edge is contained in a 4-clique.

Table 1: Upper bounds and lower bounds for connectivity problems

	Global	Single-Source	all-pairs	Steiner
edge connectivity, unweighted graphs	$\tilde{O}(m)$ [Kar00]	$\hat{O}(m)$ [AKL+22]	$\hat{O}(m)$ [AKL+22]	$\hat{O}(m)$ [LP20]
vertex connectivity, unweighted graphs with general density	$\hat{O}(m)$ [LNP+21]	$\hat{\Omega}(m^{1.5})$ for comb. algo., Corollary 3.11	$\hat{\Omega}(m^2)$ for comb. algo., Corollary 3.10	$\hat{\Omega}(m^2)$ for comb. algo., Corollary of Theorem 4.1
		$\hat{O}(m^{1.5})$, assuming Conjecture 2.5, Theorem 5.2	$\hat{O}(m^2)$, assuming Conjecture 2.5, Theorem 5.1	$\hat{O}(m^2)$, assuming Conjecture 2.5, Theorem 5.1
		$\hat{O}(m^{5/3})$, Theorem 5.2	$\hat{O}(m^{11/5})$, Theorem 5.1	$\hat{O}(m^{11/5})$, Theorem 5.1
vertex connectivity, dense unweighted graphs $m = \Theta(n^2)$	$\hat{O}(n^2)$ [LNP+21]	$\hat{\Omega}(n^3)$ for comb. algo., Corollary 3.9	$\hat{\Omega}(n^4)$ for comb. algo., Theorem 3.1	$\hat{\Omega}(n^4)$ for comb. algo. Theorem 4.1
			$\hat{\Omega}(n^{\omega(1,2,1)})$ for all algo., Remark 3.8	
		$\hat{O}(n^3)$ trivially	$\hat{O}(n^4)$ trivially	$\hat{O}(n^4)$ trivially
vertex connectivity, sparse weighted graphs, $m = \tilde{O}(n)$			$\hat{\Omega}(n^3)$ [AKT20]	

Theorem 1.3. *Given an n-vertex m-edge unweighted graph, there is an algorithm to solve the APVC problem in $\hat{O}(m^{11/5})$ time with high probability. Assuming that the* element connectivity Gomory-Hu tree *can be constructed in $\hat{O}(m)$ time, the running time can be improved to $\hat{O}(m^2)$.*

Theorem 1.3 gives the first subcubic algorithm for computing all-pairs vertex connectivity in sparse graphs. Moreover, the bound $\hat{\Theta}(m^2)$ is tight with our lower bounds for all density regimes, assuming the almost-linear-time construction of the element connectivity Gomory-Hu tree.

We note that the *element connectivity Gomory-Hu tree* is a generalization of the well-known *edge connectivity Gomory-Hu tree* [GH61], whose almost-linear-time construction was very recently shown by Abboud *et al.* [AKL+22]. It is quite believable that an element connectivity Gomory-Hu tree can also be constructed in almost-linear time.

The complexity landscape of vertex connectivity. Based on these main results, we obtain some other results and corollaries on all variants of vertex connectivity problems, as summarized in Table 1. In contrast to the edge connectivity problems which can all be solved in almost-linear time, there are three separate regimes for vertex connectivity.

1. all-pairs and Steiner vertex connectivity have complexity $\hat{\Theta}(n^4)$,
2. single-source vertex connectivity has complexity $\hat{\Theta}(n^3)$, and
3. single-source-single-sink and global vertex connectivity have complexity $\hat{\Theta}(n^2)$.

For graphs with general density, we obtain tight bounds of $\hat{\Theta}(m^2)$, $\hat{\Theta}(m^{1.5})$, $\hat{\Theta}(m)$, respectively, assuming Gomory-Hu trees for element connectivity can be computed in almost-linear time.

Discussions and Open Problems. Our lower bounds for combinatorial algorithms are particularly relevant to the context of vertex connectivity since basically all algorithms for the problems are indeed combinatorial. There are a few exceptions [LLW88, AGI+19], but these algorithms are still far from optimal (even cannot break our combinatorial lower bounds). It is a very interesting open problem whether one can bypass our combinatorial lower bounds using fast matrix multiplications, or show conditional lower bounds for general algorithms that match our combinatorial lower bounds.

1.2 Technical Overview

To prove Theorem 1.1, we are will reduce the 4-clique problem to the APVC problem. Previously, Abboud et al. [AGI+19] showed the hardness of all-pairs vertex connectivity in *directed* graphs using the 4-clique problem, which inspired our reduction. However, the techniques are not strong enough to work on undirected graphs. In more details, the hard instance of [AGI+19] is a directed acyclic graph with four layers, and so they only need to consider directed paths of length at most 3. In contrast, when we consider undirected graphs, paths connecting sources and sinks can be much more complex, which requires more advanced techniques and more careful arguments. Let us sketch our construction below.

Starting from a 4-clique instance G, it is helpful to consider its 4-partite version G_{4p}, which is simply constructed by duplicating $V(G)$ into 4 groups A, B, C, D and copying $E(G)$ for each pair of different groups (see Definition 2.2 for a formal definition). A natural way to answer the 4-clique problem on G is then checking for each pair of adjacent $a \in A$ and $d \in D$, whether there exists an adjacent pair of vertices $b \in B$ and $c \in C$ that is adjacent to both a and d (call such (b, c) a *4-clique witness* of (a, d)).

To correspond this to a vertex connectivity problem, for each pair of a and d, consider a 4-layer graph $\hat{H}_{ad}$ defined as follows. Let B_a denote the set of vertices in B adjacent to a (also define B_d, C_a and C_d similarly). Then $B_a \cap B_d$ (resp. $C_a \cap C_d$) are vertices in B (resp. C) adjacent to both a and d. The vertices of $\hat{H}_{ad}$ are $V(\hat{H}_{ad}) = \{a\} \cup (B_a \cap B_d) \cup (C_a \cap C_d) \cup \{d\}$, where the first layer (resp. the last layer) has only a single vertex a (resp. d), and the second layer (resp. the third layer) has vertices $B_a \cap B_d$ (resp. $C_a \cap C_d$). The edge set of $\hat{H}_{ad}$ is $E(\hat{H}_{ad}) = \{(a, b) \mid b \in B_a \cap B_d\} \cup E_{G_{4p}}(B_a \cap B_d, C_a \cap C_d) \cup \{(c, d) \mid c \in C_a \cap C_d\}$[4], which connects vertex a (resp. d) to each second-layer (resp. third layer) vertex, and connects the second layer $B_a \cap B_d$ and the third layer $C_a \cap C_d$ using the same edges in G_{4p}. One can simply observe that a 4-clique witness (b, c) of (a, d) exists if and only if $\kappa_{\hat{H}_{ad}}(a, d) \geq 1$[5]. To check the existence of 4-clique witnesses for all pairs (a, d) simultaneously, our final APVC instance H will be a combination of all $\hat{H}_{ad}$, and the main challenge is to make the combination compact.

We overcome this challenge by introducing two modules called the *source-sink isolating gadgets* and the *set-intersection filter*. Interestingly, our technique for proving time lower bounds is inspired by the space-lower bound techniques. More specifically, the source-sink isolating gadget is inspired by the construction in [PSY22].

The intuition behind these two modules is as follows. When considering $\kappa_{\hat{H}_{ad}}(a, d)$ of a specific pair of $a \in A$ and $d \in D$, we will somehow "remove" redundant vertices in $H \setminus \hat{H}_{ad}$ so that $\kappa_{\hat{H}_{ad}}(a, d)$ can be derived from $\kappa_H(a, d)$. The high-level idea to remove a redundant vertex v in

[4] Here $E_{G_{4p}}(B_a \cap B_d, C_a \cap C_d) \subseteq E(G_{4p})$ denote the set of edges connecting $B_a \cap B_d$ and $C_a \cap C_d$ in G_{4p}.

[5] In the overview, we use $\kappa_H(a, d)$ to denote the vertex connectivity between a and d in a graph H.

H is to create a "flow" path from a to d through v (e.g. a simple path made up of two edges (a, v) and (v, d)). By a simple flow-cut argument, this path will enforce that v appears in any (a, d)-vertex cut, while bringing some additive deviations to estimate $\kappa_{\hat{H}_{ad}}(a, d)$ as $\kappa_H(a, d)$. These modules apply this simple rule in a more general way. The source-sink isolating gadget will remove all vertices in A and D except a and d, and the intersection patterns will generate $B_a \cap B_d$ and $C_a \cap C_d$ by removing other vertices in B and C. The remaining graph will then be exactly $\hat{H}_{ad}$, and the additive deviations between $\kappa_{\hat{H}_{ad}}(a, d)$ and $\kappa_H(a, d)$ can be computed and subtracted easily.

The proof of Theorem 1.2 extends the above idea to reduce a Steiner vertex connectivity problem to an edge-universal 4-clique problem. Consider an edge-universal 4-clique instance G. Based on the above construction of H, by creating more "flow" paths, we can guarantee that the additive deviations between $\kappa_{\hat{H}_{ad}}(a, d)$ and $\kappa_H(a, d)$ for all pair of (a, d) will be the same, say a value K. Therefore, for each pair (a, d), $\kappa_H(a, d)$ is either at least $K + 1$ or equal to K, and there is no 4-clique containing a and d in the original graph G if and only if the latter case happens. Finally, checking the Steiner vertex connectivity of terminal set $A \cup D$ suffices to answer the edge-universal 4-clique problem on G.

Towards the upper bound of the APVC problem on sparse graphs in Theorem 1.3, our algorithm uses the following scheme. Let the input graph be G and let k be a degree threshold to separate vertices into two parts, the high-degree part $V_h = \{v \in V(G) \mid \deg_G(v) > k\}$ and the low-degree part $V_l = \{v \in V(G) \mid \deg_G(v) \leq k\}$. For pairs (u, v) such that $u, v \in V_h$, we can compute $\kappa_G(u, v)$ by simply calling max flows, which takes totally $O(m^2/k^2)$ calls since $|V_h| = O(m/k)$. For other pairs (u, v) with $u \in V_l$ or $v \in V_l$, there will be $\kappa_G(u, v) \leq k$, because the vertex connectivity of u and v is upper bounded by their degrees. The vertex connectivity oracle in [PSY22] can exactly capture all-pairs vertex connectivity bounded by k, which takes $\tilde{O}(k^2)$ black-box calls to Gomory-Hu trees for element connectivity whose construction time is currently $\hat{O}(mk)$. The whole algorithm takes $\hat{O}(m^{11/5})$ time by choosing a proper k, and the running time will be immediately improved to $\hat{O}(m^2)$ if Gomory-Hu trees for element connectivity can be constructed in almost linear time.

1.3 Organization

We will start with some basic notations and introduce conjectures in Section 2. In Section 3, we will show the conditional lower bound of the APVC problem. In Section 4, we will extend the approach in Section 3 to show the conditional lower bound of the Steiner vertex connectivity problem. In Section 5, we will show a simple APVC algorithm for graphs with general density.

2 Preliminaries

In this paper, we use standard graph theoretic notations. For a graph G, we use $V(G)$ and $E(G)$ to denote its vertex set and edge set. For any vertex set $V' \subseteq V(G)$, we let $G[V']$ denote the subgraph induced by V' and let $G \setminus V'$ be a short form of $G[V(G) \setminus V']$. For two graphs G_1 and G_2 which vertex sets $V(G_1)$ and $V(G_2)$ may intersect, we let $G_1 \cup G_2$ denote the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$. For two subset of vertices $V_1, V_2 \subseteq V(G)$, we let $E_G(V_1, V_2)$ denote the set of edges directly connecting V_1 and V_2. For a vertex $v \in G$, we let $N_G(v) = \{u \mid (u, v) \in E(G)\}$ denote its neighbor set, and let $\bar{N}_G(v) = V(G) \setminus N_G(v)$ denote its non-neighbors.

Vertex connectivity. In a graph G, the vertex connectivity for two vertices $u, v \in V(G)$, denoted by $\kappa_G(u, v)$, is the maximum number of internally vertex-disjoint paths from u to v. By Menger's theorem, $\kappa_G(u, v)$ is equal to the size of minimized subsets $C \subseteq (V(G) \setminus \{u, v\}) \cup E(G)$ deleting which from G will disconnect u and v.

Vertex Connectivity problems. In this paper, we will consider four vertex connectivity problems, i.e. the global, single-source, all-pairs, and Steiner vertex connectivity problems. The edge connectivity versions of these problems are analogous.

- **The global vertex connectivity problem.** Given an undirected unweighted graph G, the global vertex connectivity problem (the global-VC problem) asks the global vertex connectivity, denoted by κ_G, where $\kappa_G = \min_{u,v \in G} \kappa_G(u, v)$.
- **The single-source vertex connectivity problem.** Given an undirected unweighted graph G with a source vertex s, a single source vertex connectivity problem (the SSVC problem) asks $\kappa_G(s, v)$ for all other vertices $v \in G$.
- **The all-pairs vertex connectivity problem.** Given an undirected unweighted graph G, the all-pairs vertex connectivity problem (the APVC problem) asks $\kappa_G(u, v)$ for all pairs of $u, v \in G$.
- **The Steiner vertex connectivity problem.** Given an undirected unweighted graph G with a terminal set $T \subseteq V(G)$, the Steiner vertex connectivity problem (the Steiner-VC problem) asks the Steiner vertex connectivity of T, denoted by $\kappa_G(T)$, where $\kappa_G(T) = \min_{u,v \in T} \kappa_G(u, v)$.

The 4-clique conjecture. Given an n-vertex undirected graph G, the k-clique problem is to decide whether there is a clique with k vertices in G. The k-clique problem can be solved in $O(n^k)$ time trivially, and a more efficient combinatorial algorithm takes running time $O(n^k/\log^k n)$ [Vas09]. The popular k-clique conjecture (see e.g. [BGL17]) suggests that there is no combinatorial algorithm for the k-clique problem with $O(n^{k-\epsilon})$ running time for any constant $\epsilon > 0$. In Section 3, we will use this conjecture in the case $k = 4$.

Conjecture 2.1 (4-clique conjecture)**.** *There is no combinatorial algorithm that solves the 4-clique problem for n-vertex graphs in $O(n^{4-\epsilon})$ time for any constant $\epsilon > 0$.*

For each 4-clique instance G, it is equivalent to consider its 4-partite form G_{4p} as defined below. Note that a 4-clique in a 4-partite graph should contain exactly one vertex from each group, and the original graph has a 4-clique if and only if the 4-partite graph G_{4p} has a 4-clique.

Definition 2.2 (4-partite graph G_{4p})**.** Given an undirected graph G, the 4-partite graph G_{4p} of G has vertex set $V(G_{4p}) = \{v_A, v_B, v_C, v_D \mid v \in V(G)\}$, and we let $A = \{v_A \mid v \in V(G)\}, B = \{v_B \mid v \in V(G)\}, C = \{v_C \mid v \in V(G)\}, D = \{v_D \mid v \in V(G)\}$ be four groups partitioning $V(G_{4p})$. The edge set $E(G_{4p}) = \{(u_X, v_Y) \mid (u, v) \in E(G), X, Y \in \{A, B, C, D\}, X \neq Y\}$.

The edge-universal 4-clique conjecture. We consider a variant of the 4-clique problem, called the *edge-universal 4-clique problem*. Given an undirected graph G and a subset of *demand* edges $E_{\text{dem}} \subseteq E(G)$, this problem asks if every edge in E_{dem} is contained by a 4-clique. We suggest the following conjecture on this problem.

Conjecture 2.3 (Edge-universal 4-clique conjecture)**.** *There is no combinatorial algorithm that answers the all edge 4-clique problem for n-vertex graphs in $O(n^{4-\epsilon})$ time for any constant $\epsilon > 0$.*

We note our formulation of the edge-universal 4-clique problem is at least as hard as the problem of checking if every edge is contained in some 4-clique by fixing $E_{\text{dem}} = E(G)$. We choose to present this formulation that allows any $E_{\text{dem}} \subseteq E(G)$ because it only strengthens our hardness result and shows the flexibility of our reduction.

The difference between the edge-universal 4-clique conjecture vs. the 4-clique conjecture is that we switch the quantifier in the definition of the problems. This allows us to obtain new hardness results. This strategy for proving conditional lower bounds has been studied and done several times in the literature of fine-grained complexity [GIKW19, AWW16, ABHS22]. For example, the relationship between the edge-universal 4-clique problem vs. the 4-clique problem is the same as the relationship between the well-known *orthogonal vector* problem vs. the *hitting set* problem introduced in [AWW16], and *SETH* vs. *quantified SETH* introduced in [ABHS22].

Gomory-Hu trees for element connectivity Gomory-Hu trees are cut-equivalent trees introduced by Gomory and Hu [GH61] to *capture* all-pairs edge connectivity in weighted graphs. More precisely, given a Gomory-Hu tree, the edge connectivity of any given pair of vertices can be queried in nearly constant time. Very recently, a breakthrough by [AKL+22] showed that a Gomory-Hu tree can be constructed in $\tilde{O}(n^2)$ time for a weighted graph and $\hat{O}(m)$ time for an unweighted graph. For vertex connectivity, it has been shown by [Ben95] that there are no such cut-equivalent trees. See also [PSY22] for a more general space lower bound forbidding the existence of such trees for vertex connectivity.

Element connectivity is the notion of connectivity that is related to vertex connectivity, and yet Gomory-Hu trees have been shown to exist for element connectivity (see e.g. [CRX15]). Given an undirected graph G and a terminal set $U \subseteq V(G)$, the element connectivity for two vertices $u, v \in U$, denoted by $\kappa'_{G,U}(u, v)$, is the size of minimized set $C \subseteq E(G) \cup (V(G) \setminus U)$ whose removal will disconnect u and v. An element connectivity Gomory-Hu tree for the graph G and terminal set U will capture $\kappa'_{G,U}(u, v)$ for all pairs of $u, v \in U$.

In [PSY22], they consider a variant called k-Gomory-Hu tree for element connectivity, which given an additional parameter k, will capture the value $\min\{\kappa'_{G,U}(u, v), k\}$ for all $u, v \in U$ (namely we can query $\min\{\kappa'_{G,U}(u, v), k\}$ for each given $u, v \in U$ in nearly constant time). By generalizing the $(1 + \epsilon)$-approximate Gomory-Hu tree algorithm by [LP21] to the element connectivity setting, the following result was obtained by [PSY22].

Theorem 2.4. *Given an n-vertex m-edge undirected unweighted graph G, a terminal set $U \subseteq V(G)$ and a parameter k, there is a randomized algorithm to construct a k-Gomory-Hu tree for element connectivity in $\hat{O}(mk)$ time with high probability.*

Given the similarity of Gomory-Hu trees for edge connectivity and element connectivity, and the recent breakthrough of $\hat{O}(m)$-time construction algorithm for edge connectivity Gomroy-Hu tree, it seems reasonable to conjecture that element connectivity Gomory-Hu tree can also be constructed in $\hat{O}(m)$ time.

Conjecture 2.5. *Given an n-vertex m-edge undirected unweighted graph G and a terminal set $U \subseteq V(G)$, an element connectivity Gomory-Hu tree can be constructed in $\hat{O}(m)$ time.*

We leave this conjecture as a very interesting open problem.

3 The Lower Bound for the APVC Problem

In this section, we will prove Theorem 3.1, a conditional lower bound of the APVC problem in undirected unweighted graphs conditioning on the 4-clique conjecture. Concretely, we will show a reduction from the 4-clique problem to the APVC problem.

Theorem 3.1. *Assuming Conjecture 2.1, for n-vertex undirected unweighted graphs, there is no combinatorial algorithm that solves the APVC problem in $O(n^{4-\epsilon})$ time for any constant $\epsilon > 0$.*

Given an n-vertex 4-clique instance G, let G_{4p} be the corresponding 4-partite graph defined in Definition 2.2 (where $V(G_{4p})$ is partitioned into 4 groups A, B, C, D). We start with some notations. For each vertex $a \in A$, we use $B_a = \{b \in B | (a, b) \in E(G_{4p})\}$ to denote the neighbors of a in B and let $\bar{B}_a = B \setminus B_a$. Analogously, C_a is the set of vertices in C adjacent to a and $\bar{C}_a = C \setminus C_a$. For each $d \in D$, we define $B_d, \bar{B}_d, C_d, \bar{C}_d$ in a similar way.

As discussed in Section 1.2, our 4-clique instance H will be constructed using two kinds of modules, the source-sink isolating gadgets and the set-intersection filters, which will be introduced in Section 3.1 and Section 3.2 respectively. After that, the final construction of H and the proof of Theorem 3.1 will be completed in Section 3.3.

3.1 The Source-Sink Isolating Gadget

We first introduce the *source-sink isolating gadget*. Basically, for an undirected graph R and two disjoint groups of vertices $X, Y \subseteq V(R)$, a source-sink isolating gadget $Q(X, Y)$ (or just Q for short) is a graph on vertices $X \cup Y$ with additional vertices outside R. Its formal guarantee is as follows.

Lemma 3.2. *Given an undirected graph R and two disjoint groups of vertices $X, Y \subseteq V(R)$, there is a graph Q with $V(Q) \cap V(R) = X \cup Y$ and $|V(Q)| = O(|X| + |Y|)$ such that for any $x \in X, y \in Y$ with $(x, y) \notin E(R)$,*

$$\kappa_{R \cup Q}(x, y) = \kappa_{R_{xy}}(x, y) + |X| + |Y|,$$

where $R_{xy} = R \setminus ((X \cup Y) \setminus \{x, y\})$. Such a graph Q is called a source-sink isolating gadget, and moreover, the construction of Q is independent from R.

The reason we call the graph Q a source-sink isolating gadget is that by adding Q into the input graph R the vertex connectivity between any pair of source $x \in X$ and sink $y \in Y$ in $R \cup Q$, i.e., $\kappa_{R \cup Q}(x, y)$, can be derived from their connectivity in R_{xy}, i.e., $\kappa_{R_{xy}}(x, y)$. But the graph R_{xy}, as defined in Lemma 3.2, is just the graph R after removing all source and sink vertices in X and Y except x and y. That is, the gadget "isolates" the pair x and y from the rest. We will use this gadget in Section 3.3.

Proof. We construct Q in the following way. We create duplicated sets X_1, X_2 of X, and also Y_1, Y_2 of Y. For each vertex $x \in X$, we let $\hat{x}_1 \in X_1$ and $\hat{x}_2 \in X_2$ denote copies of x in X_1 and X_2

respectively if there is no other specification (for each $y \in Y$, $\hat{y}_1, \hat{y}_2$ are defined similarly). The vertex set of Q is $V(Q) = X \cup X_1 \cup X_2 \cup Y \cup Y_1 \cup Y_2$, and the edge set is

$$\begin{aligned} E(Q) = &\{(x, \hat{x}_1) \mid x \in X\} \cup \{(x_1, y) \mid x_1 \in X_1, y \in Y\} \cup \\ &\{(y, \hat{y}_1) \mid y \in Y\} \cup \{(y_1, x) \mid y_1 \in Y_1, x \in X\} \cup \\ &\{(x, x_2) \mid x \in X, x_2 \in X_2\} \cup \\ &\{(y, y_2) \mid y \in Y, y_2 \in Y_2\}. \end{aligned}$$

The construction of Q is illustrated in Figure 1.

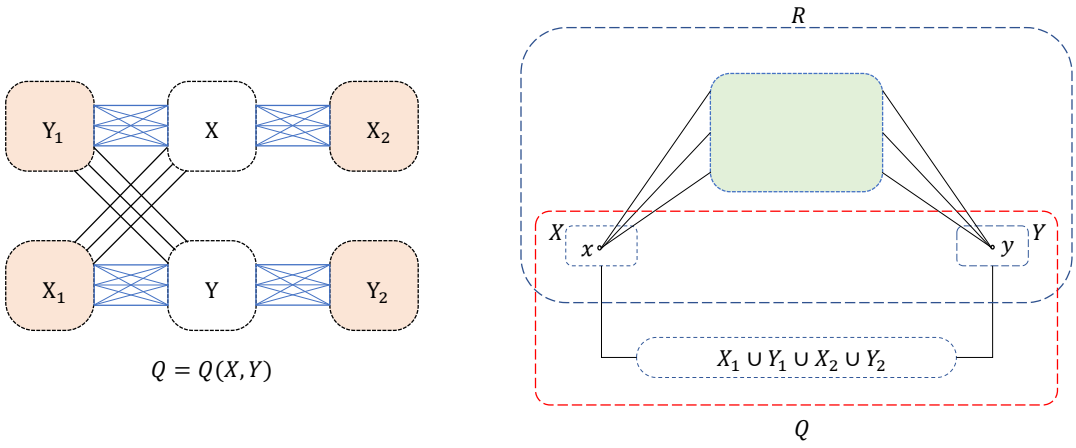

Figure 1: The source-sink isolating gadget Q and the whole graph $R \cup Q$. There are bipartite cliques between X and Y_1, X_2, as well as Y and X_1, Y_2, and there are perfect matchings between X and X_1, Y and Y_1.

Fixing some $x \in X$ and $y \in Y$, we let $\kappa = \kappa_{R_{xy}}(x, y)$ for short. We first show $\kappa_{R \cup Q}(x, y) \geq \kappa + |X| + |Y|$. From the flow view of vertex connectivity, there are κ internal vertex-disjoint paths from x to y in R_{xy}. Combining Claim 3.3 and $V(Q) \cap V(R_{xy}) = \{x, y\}$, there are $\kappa + |X| + |Y|$ internal vertex-disjoint paths in $R \cup Q$. Therefore, $\kappa_{R \cup Q}(x, y) \geq \kappa + |X| + |Y|$.

Claim 3.3. *There are $|X| + |Y|$ internal vertex-disjoint paths from x to y in Q.*

Proof. The first path is $x \to \hat{x}_1 \to y$. Each of the next $|X| - 1$ paths corresponds to each $x' \in X$ s.t. $x' \neq x$, the path $x \to \hat{x}'_2 \to x' \to \hat{x}'_1 \to y$ concretely, where $\hat{x}'_1$ and $\hat{x}'_2$ are copies of x' in X_1 and X_2. Symmetrically, there is a path $x \to y_1 \to y$ and $|Y| - 1$ paths, each of which corresponds to each $y' \in Y$ s.t. $y' \neq y$ (namely the path $x \to \hat{y}'_1 \to y' \to \hat{y}'_2 \to y$, where $\hat{y}'_1$ and $\hat{y}'_2$ are copies of y' in Y_1 and Y_2). Observe that these $|X| + |Y|$ paths are internal vertex-disjoint. □

We then argue from the cut view that $\kappa_{R\cup Q}(x,y) \le \kappa + |X| + |Y|$. Consider the vertex set $S_Q = \{x' \mid x' \in X, x' \neq x\} \cup \{y' \mid y' \in Y, y' \neq y\} \cup \{\hat{x}_1, \hat{y}_1\}$. After removing S_Q from $R \cup Q$, observe that vertices in both R and Q are only x and y, so a simple path from x to y in graph $(R \cup Q) \setminus S_Q$ will be totally inside subgraphs $Q \setminus S_Q$ or $R \setminus S_Q$. Note that x and y are disconnected in $Q \setminus S_Q$. Moreover, subgraph $R \setminus S_Q$ is exactly R_{xy}, so removing κ vertices can disconnect x and y in $R \setminus S_Q$. In conclusion, in graph $R \cup Q$, x and y can be disconnected by removing $|S_Q| + \kappa$ vertices, so $\kappa_{R\cup Q}(x,y) \le \kappa + |X| + |Y|$.

Finally, the size of Q follows directly from the construction.

□

3.2 The Set-Intersection Filter

We now introduce the set-intersection filter. For each $a \in A$, $d \in D$, the set-intersection filter P^B_{ad} is a subgraph of the final H, which will "filter" the intersection $B_a \cap B_d$ from the whole set B as Lemma 3.4 shows. It is constructed as follows. Let $V(P^B_{ad}) = \{a\} \cup B \cup B' \cup \{d\}$, where B' duplicates vertices in B. For each vertex $b \in B$, we use $\hat{b}'$ to denote its copy in B', and for each (non-)neighbor sets $B_a, \bar{B}_a, B_d, \bar{B}_d \subseteq B$, we use $B'_a, \bar{B}'_a, B'_d, \bar{B}'_d \subseteq B'$ to denote their counterparts respectively. The edge set of P^B_{ad} is constructed by

$$\begin{aligned} E(P^B_{ad}) =& \{(a,b) \mid b \in B\} \cup \{(b,d) \mid b \in \bar{B}_d\} \cup \\ & \{(a,\hat{b}') \mid b \in B_a\} \cup \{(b',d) \mid b' \in B'\} \cup \\ & \{(b,\hat{b}') \mid b \in B\}. \end{aligned}$$

See Figure 2 for an illustration of P^B_{ad}.

The construction of $E(P^B_{ad})$ can be interpreted in the following intuitive way.

- First, the edges $\{(a,b) \mid b \in B\}$ and $\{(b,d) \mid b \in \bar{B}_d\}$ create vertex-disjoint paths of the format $a \to b \to d$ for all $b \in \bar{B}_d$, which implies $\bar{B}_d$ will be cut from B in every vertex cut of (a,d).
- Second, the edges $\{(a,\hat{b}') \mid b \in B_a\}$ and $\{(b',d) \mid b' \in B'\}$ create vertex-disjoint paths of the format $a \to \hat{b}' \to d$ for all $b \in B_a$, which analogously implies that B'_a will be cut from B' in every (a,d)-vertex cut.
- Third, for every (a,d)-vertex cut, after $\bar{B}_d$ and B'_a are cut from B and B' respectively from the above discussion, either b or $\hat{b}'$ should be cut for all $b \in B_d \cap \bar{B}_a$. The reason is that the edges $\{(b,\hat{b}') \mid b \in B\}$ form a matching between B and B', which will create vertex-disjoint paths of the format $a \to b \to \hat{b}' \to d$ for all $b \in B_d \cap \bar{B}_a$.

Therefore, suppose that in the third step we choose to cut vertex b of all $b \in B_d \cap \bar{B}_a$. (In the formal proof of Lemma 3.4, we will see that, cutting b rather than $\hat{b}'$ for all $b \in B_d \cap \bar{B}_a$ is always a better choice when considering vertex min cut between a and d.) Now $\bar{B}_d$ is cut in the first step and $B_d \cap \bar{B}_a$ is cut in the third step, so vertices in B that survive are $B \setminus \bar{B}_d \setminus (B_d \cap \bar{B}_a) = B_a \cap B_d$. Therefore, the set-intersection filter indeed obtain $B_a \cap B_d$ as desired.

Lemma 3.4. *For each $a \in A, d \in D$, the set-intersection filter P^B_{ad} has the following property. Let R be an undirected graph such that $V(R) \cap V(P^B_{ad}) = \{a\} \cup B \cup \{d\}$, then*

$$\kappa_{R\cup P^B_{ad}}(a,d) = \kappa_{R^B_{ad}}(a,d) + |\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a|,$$

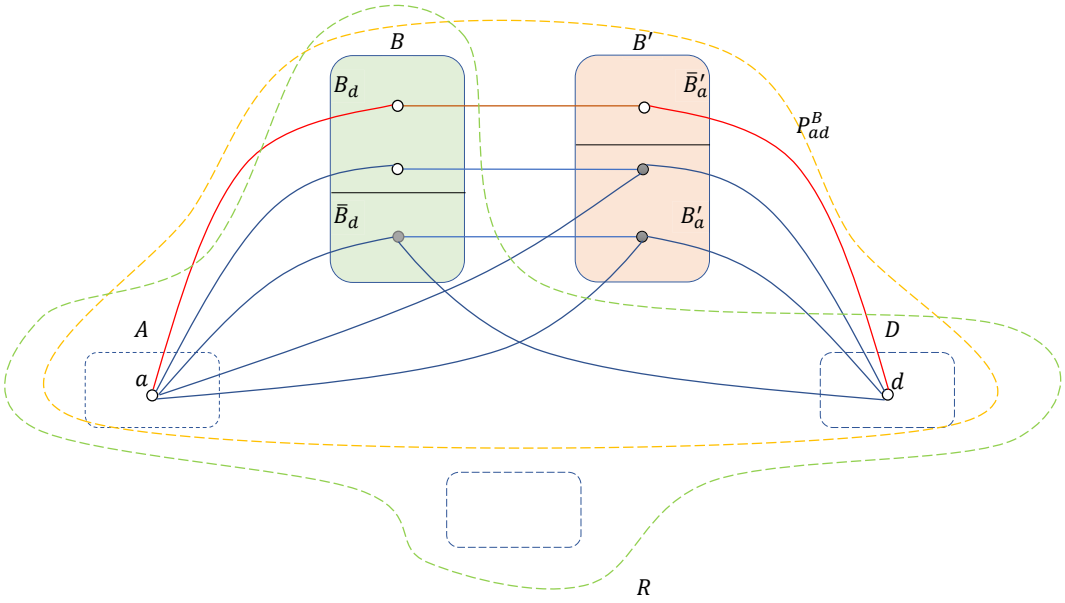

Figure 2: The set-intersection filter. The sets P^B_{ad} and R are the areas surrounded by dotted lines.

where $R^B_{ad} = (R \cup P^B_{ad}) \setminus ((B \setminus (B_a \cap B_d)) \cup B')$ *equivalent to* $(R \setminus (B \setminus (B_a \cap B_d))) \cup \{(a, b) \mid b \in B_a \cap B_d\}$, *that is, the graph starting from* R, *removing non* $B_a \cap B_d$ *vertices and then adding edges connecting* a *and each* $b \in B_a \cap B_d$.

Proof. Let $\kappa = \kappa_{R^B_{ad}}(a, d)$ for short. We first show $\kappa_{R \cup P^B_{ad}}(a, d) \geq \kappa + |\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a|$. There are κ internal vertex-disjoint paths from a to d in R^B_{ad}. Because $V(P^B_{ad}) \cap V(R^B_{ad}) = (B_a \cap B_d) \cup \{a, d\}$, the paths from a to d in $P^B_{ad} \setminus (B_a \cap B_d)$ are internal disjoint with those in R^B_{ad}, and there are $|\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a|$ of them by Claim 3.5. Therefore, we have $\kappa + |\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a|$ internal vertex-disjoint paths from a to d in $R \cup P^B_{ad}$.

Claim 3.5. *There are* $|\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a|$ *internal vertex-disjoint paths from* a *to* d *in* $P^B_{ad} \setminus (B_a \cap B_d)$.

Proof. The first $|\bar{B}_d|$ paths correspond to vertices $b \in \bar{B}_d$, each of which has a path $a \to b \to d$. The next $|B_a|$ paths correspond to vertices $b \in B_a$, each of which has a path $a \to \hat{b}' \to d$. The last $|B_d \cap \bar{B}_a|$ paths correspond to vertices $b \in B_d \cap \bar{B}_a$, each of which has a path $a \to b \to \hat{b}' \to d$. □

We then complete the proof by showing $\kappa_{R \cup P^B_{ad}}(a, d) \leq \kappa + |\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a|$. Let $S_P = \{b \in B \mid b \in \bar{B}_d\} \cup \{\hat{b}' \in B' \mid b \in B_a\} \cup \{b \in B \mid b \in B_d \cap \bar{B}_a\}$ be a vertex cut of (a, d) in P^B_{ad}. Observe that in $(R \cup P^B_{ad}) \setminus S_P$, a path from vertex a to a vertex $b' \in B' \setminus S_P$ must go through vertex d, because each $b' \in B' \setminus S_P$ only connects to d after removing S_P. Therefore, it is safe to ignore $B' \setminus S_P$ when considering the vertex connectivity between a and d in graph $(R \cup P^B_{ad}) \setminus S_P$, i.e.

$$\kappa_{(R \cup P^B_{ad}) \setminus S_P}(a, d) = \kappa_{(R \cup P^B_{ad}) \setminus (S_P \cup B')}(a, d) = \kappa_{R^B_{ad}}(a, d) = \kappa,$$

which means by further removing κ vertices, we can disconnect a and d in $(R \cup P^B_{ad}) \setminus S_P$.

In conclusion, we can remove $|S_P| + \kappa$ vertices to disconnect a and d in $R \cup P^B_{ad}$, which implies $\kappa_{R \cup P^B_{ad}}(a, d) \leq \kappa + |\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a|$.

□

For each $a \in A, d \in D$, we also define the set-intersection filter P^C_{ad} similarly but symmetrically. Let $V(P^C_{ad}) = \{a\} \cup C' \cup C \cup \{d\}$, where C' duplicates vertices in C. For each $c \in C$, let $\hat{c}'$ denote its copy in C'. For each (non-)neighbor sets $C_a, \bar{C}_a, C_d, \bar{C}_d \subseteq C$, let $C'_a, \bar{C}'_a, C'_d, \bar{C}'_d \subseteq C'$ denote their counterparts respectively. The edge set is

$$\begin{aligned} E(P^C_{ad}) = &\{(a, c) \mid c \in \bar{C}_a\} \cup \{(c, d) \mid c \in C\} \cup \\ &\{(a, c') \mid c' \in C'\} \cup \{(\hat{c}', d) \mid c \in C_d\} \cup \\ &\{(\hat{c}', c) \mid c \in C\}. \end{aligned}$$

The pattern P^C_{ad} will also have similar properties as shown below.

Lemma 3.6. *For each $a \in A, d \in D$, the set-intersection filter P^C_{ad} has the following property. Let R be an undirected graph such that $V(R) \cap V(P^C_{ad}) = \{a\} \cup C \cup \{d\}$, then*

$$\kappa_{R \cup P^C_{ad}}(a, d) = \kappa_{R^C_{ad}}(a, d) + |C_d| + |\bar{C}_a| + |C_a \cap \bar{C}_d|,$$

where $R^C_{ad} = (R \cup P^C_{ad}) \setminus ((C \setminus (C_a \cap C_d)) \cup C')$ equivalent to $(R \setminus (C \setminus (C_a \cap C_d))) \cup \{(c, d) \mid c \in C_a \cap C_d\}$, that is, the graph starting from R, removing non $C_a \cap C_d$ vertices and then adding edges connecting d and each $c \in C_a \cap C_d$.

3.3 The Final Construction of the APVC Instance

We are now ready to construct the final APVC instance H. For each $a \in A, d \in D$, we first construct a graph H_{ad} as follows. Let P^B_{ad} and P^C_{ad} be the set-intersection filters defined in Section 3.2. Then the graph H_{ad} will be defined by

$$H_{ad} = P^B_{ad} \cup P^C_{ad} \cup G_{4p}[B \cup C],$$

which is the union of two set-intersection filters with edges in the graph G_{4p} connecting B and C. The final graph then will be constructed by

$$H = \bigcup_{a \in A, d \in D} H_{ad} \cup Q(A, D),$$

where $Q(A, D)$ is the source-sink isolating gadget from Lemma 3.2 given $R = \bigcup_{a \in A, d \in D} H_{ad}$ and the sets A, D. See Figure 3 below for an illustration.

Lemma 3.7. *For each $a \in A$ and $d \in D$ of G_{4p}, we have*

$$\kappa_H(a, d) \geq 4n + |N_G(a) \cap \bar{N}_G(d)| + |\bar{N}_G(a) \cap N_G(d)|. \tag{1}$$

Furthermore, without ambiguity let a and d also denote their original vertices in G. Then there is a 4-clique in G containing a and d if and only if a and d are adjacent in G_{4p} and

$$\kappa_H(a, d) \geq 4n + |N_G(a) \cap \bar{N}_G(d)| + |\bar{N}_G(a) \cap N_G(d)| + 1. \tag{2}$$

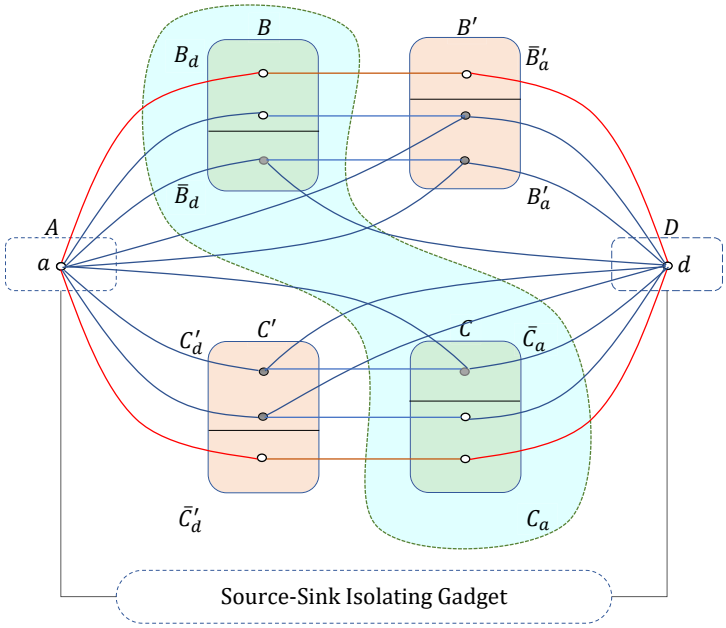

Figure 3: The graph H. The edges connecting B and C (i.e. $G_{4p}[B \cup C]$) are omitted in the highlighted part.

Proof. Fixing some $a \in A, d \in D$, we first apply Lemma 3.2 on H and vertex sets A, D, which gives $\kappa_H(a,d) = \kappa_{R_{ad}}(a,d) + |A| + |D|$, where $R_{ad} = (\bigcup_{a\in A, d\in D} H_{ad}) \setminus ((A \cup D) \setminus \{a,d\})$. In fact, the graph R_{ad} is exactly H_{ad}, because the induced subgraphs $H_{ad}[B \cup B' \cup C \cup C']$ are the same for all $a \in A, d \in D$ by construction. Therefore, we have

$$\kappa_H(a,d) = \kappa_{H_{ad}}(a,d) + |A| + |D|. \tag{3}$$

Recall that $H_{ad} = P^B_{ad} \cup P^C_{ad} \cup G_{4p}[B \cup C]$. Let $R_1 = P^C_{ad} \cup G_{4p}[B \cup C]$. We apply Lemma 3.4 on P^B_{ad} and R_1, which gives

$$\kappa_{H_{ad}}(a,d) = \kappa_{R_1 \cup P^B_{ad}}(a,d) = \kappa_{R'_1}(a,d) + |\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a|, \tag{4}$$

where

$$\begin{aligned} R'_1 &= (R_1 \setminus (B \setminus (B_a \cap B_d))) \cup \{(a,b) \mid b \in B_a \cap B_d\} \\ &= P^C_{ad} \cup G_{4p}[(B_a \cap B_d) \cup C] \cup \{(a,b) \mid b \in B_a \cap B_d\}. \end{aligned}$$

Let $R_2 = G_{4p}[(B_a \cap B_d) \cup C] \cup \{(a,b) \mid b \in B_a \cap B_d\}$. We apply Lemma 3.6 on P^C_{ad} and R_2, which gives

$$\kappa_{R'_1}(a,d) = \kappa_{R_2 \cup P^C_{ad}}(a,d) = \kappa_{R'_2}(a,d) + |C_d| + |\bar{C}_a| + |C_a \cap \bar{C}_d|, \tag{5}$$

where

$$\begin{aligned} R'_2 =& (R_2 \setminus (C \setminus (C_a \cap C_d))) \cup \{(c,d) \mid c \in C_a \cap C_d\} \\ =& G_{4p}[(B_a \cap B_d) \cup (C_a \cap C_d)] \cup \{(a,b) \mid b \in B_a \cap B_d\} \cup \\ & \{(c,d) \mid c \in C_a \cap C_d\}. \end{aligned}$$

We use $\hat{H}_{ad}$ to denote R'_2 in the remaining proof and note that it is equivalent to the definition of $\hat{H}_{ad}$ in Section 1.2. Because $\kappa_{\hat{H}_{ad}}(a,d) \geq 0$, combining Equations (3) to (5), we get

$$\kappa_H(a,d) \geq |A| + |D| + |\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a| + |C_d| + |\bar{C}_a| + |C_a \cap \bar{C}_d|. \tag{6}$$

We now prove the second part of the lemma. If a and d are not adjacent in G_{4p}, they are not adjacent in G either, so there is no 4-clique in G containing them. Otherwise, a and d are adjacent, we claim that there is a 4-clique containing a and d in G_{4p} (which is equivalent to the existence of a 4-clique containing a and d in G by Definition 2.2) if and only if $\kappa_{\hat{H}_{ad}}(a,d) \geq 1$. If the 4-clique exists, let $b \in B$ and $c \in C$ be the other vertices in the 4-clique. Then there is a path $a \to b \to c \to d$ in $\hat{H}_{ad}$ from the construction, which implies $\kappa_{\hat{H}_{ad}}(a,d) \geq 1$. On the other hand, if $\kappa_{\hat{H}_{ad}}(a,d) \geq 1$, there is a path $a \to b \to c \to d$, and (a,b,c,d) forms a 4-clique in G_{4p}. Combining this claim with Equations (3) to (5) gives that there is a 4-clique containing a and d in G if and only if

$$\kappa_H(a,d) \geq |A| + |D| + |\bar{B}_d| + |B_a| + |B_d \cap \bar{B}_a| + |C_d| + |\bar{C}_a| + |C_a \cap \bar{C}_d| + 1. \tag{7}$$

Finally, by the construction of G_{4p}, we have $|A| = |B| = |C| = |D| = n$, $|B_a| = |C_a|$, $|B_d| = |C_d|$, $|B_d \cap \bar{B}_a| = |N_G(d) \cap \bar{N}_G(a)|$ and $|C_a \cap \bar{C}_d| = |N_G(a) \cap \bar{N}_G(d)|$. Combining them with Inequalities (6) and (7) completes the proof.

□

Proof of Theorem 3.1. Assume for contradiction that there exists a combinatorial algorithm $\mathcal{A}$ for the APVC problem with running time $O(n^{4-\epsilon})$ for some constant $\epsilon > 0$. Let G be an arbitrary 4-clique instance. We first construct the 4-partite graph G_{4p} and the graph H following the construction in this section. Note that $V(H) = V(Q) \cup A \cup B \cup B' \cup C \cup C' \cup D$, so $|V(H)| = O(n)$ by the construction and Lemma 3.2. Also, H can be constructed in $O(n^2)$ time directly. By Lemma 3.7, we can solve the 4-clique problem by first running $\mathcal{A}$ on graph H and then checking for each adjacent $a \in A, d \in D$ whether $\kappa_H(a,d)$ reaches the threshold value $4n + |N_G(a) \cap \bar{N}_G(d)| + |\bar{N}_G(a) \cap N_G(d)| + 1$. This takes $O(n^{4-\epsilon}) + O(n^3) = O(n^{4-\epsilon})$ time because computing the threshold value for each $a \in A, d \in D$ takes totally $O(n^3)$ extra time. This contradicts Conjecture 2.1.

□

Remark 3.8. The proof of Theorem 3.1 basically shows that if the APVC problem can be solved in time $T_{\text{APVC}}(n)$, then the 4-clique problem can be solved in time $T_{\text{4clique}}(n) = O(T_{\text{APVC}}(n) + n^3)$. We note that this relation can be improved to $T_{\text{4clique}}(n) = O(T_{\text{APVC}}(n) + n^\omega)$ if we use fast matrix multiplication to speed up the reduction. For general algorithms, another version of the 4-clique conjecture (see e.g. [DW22]) states that solving the 4-clique problem requires $n^{\omega(1,2,1)-o(1)}$ time. Therefore, assuming this conjecture, solving the APVC problem also requires $n^{\omega(1,2,1)-o(1)}$ time for general algorithms.

3.4 Further Results

In this section, we will show several corollaries from the lower bound of the APVC problem.

The first corollary is a conditional lower bound of the SSVC problem.

Corollary 3.9. *Assuming Conjecture 2.1, for n-vertex undirected unweighted graphs, there is no combinatorial algorithm that solves the SSVC problem in $O(n^{3-\epsilon})$ time for any constant $\epsilon > 0$.*

Proof. Assume for the contradiction that the SSVC problem can be solved in $O(n^{3-\epsilon})$ time. Then for an APVC instance G, we may treat every vertex in G as a source to get the correct output. It takes $O(n)$ SSVC calls and therefore the complexity is $O(n \cdot n^{3-\epsilon}) = O(n^{4-\epsilon})$, which contradicts Theorem 3.1 assuming Conjecture 2.1. □

The second corollary is a conditional lower bound of the APVC problem for graphs with general density.

Corollary 3.10. *Given any constant $\delta \in [0, 1]$, assuming Conjecture 2.1, there is no combinatorial algorithm that solves the APVC problem for n-vertex m-edge unweighted graphs, where $m = \Theta(n^{1+\delta})$ with running time $O(m^{2-\epsilon})$ for any constant $\epsilon > 0$.*

Proof. Assume that for some constants δ and ϵ, such $O(m^{2-\epsilon})$-time algorithm $\mathcal{A}$ exists. Let H be an $\hat{n}$-vertex $\hat{m}$-edge APVC hard instance with $\hat{m} = \Theta(\hat{n}^2)$, constructed as above for some 4-clique instance. Let G be the union of H and $\Theta(\hat{m}^{1/(1+\delta)})$ isolated vertices. Observe that G now has $n = \hat{n} + \Theta(\hat{m}^{1/(1+\delta)})$ vertices and $m = \hat{m}$ edges, i.e. $m = \Theta(n^{1+\delta})$. By applying algorithm $\mathcal{A}$ on G, the all-pairs vertex connectivity of H can be computed in $O(m^{2-\epsilon})$, i.e. $O(\hat{n}^{4-2\epsilon})$ time, contradicting Conjecture 2.1 by the argument in the proof of Theorem 3.1. □

The last corollary is a conditional lower bound of the SSVC problem for graphs with general density. The proof is omitted since it is analogous to Corollary 3.10.

Corollary 3.11. *Given any constant $\delta \in [0, 1]$, assuming Conjecture 2.1, there is no combinatorial algorithm that solves the SSVC problem for n-vertex m-edge unweighted graphs, where $m = \Theta(n^{1+\delta})$, with running time $O(m^{3/2-\epsilon})$ for any constant $\epsilon > 0$.*

4 The Lower Bound for Steiner Vertex Connectivity Problem

In this section, we will prove Theorem 4.1, a conditional lower bound of the Steiner vertex connectivity problem in undirected unweighted graphs, conditioning on the edge-universal 4-clique problem.

Theorem 4.1. *For n-vertex undirected unweighted graphs, assuming Conjecture 2.3, there is no combinatorial algorithm that solves the Steiner vertex connectivity problem in $O(n^{4-\epsilon})$ time for any constant $\epsilon > 0$.*

Given an n-vertex edge-universal 4-clique instance G with a set E_{dem} of *demand* edges, let H be the hard APVC instance we construct in Section 3. We will strengthen H to another graph J such that the edge-universal 4-clique problem in G can be reduced to a Steiner vertex connectivity problem in J of the terminal set $A \cup D$. As mentioned in Section 1.2, the main idea is to add extra "flow" paths from A to D to make the additive deviations between $\kappa_J(a, d)$ and $\kappa_{\hat{H}_{ad}}(a, d)$ uniform for all pairs of $a \in A$ and $d \in D$. Furthermore, to exclude the interference of vertex connectivity of pairs (a_1, a_2) s.t. $a_1, a_2 \in A$ or (d_1, d_2) s.t. $d_1, d_2 \in D$, we artificially add large additive deviations for these pairs.

The construction of J is as follows and see Figure 4 for an illustration. The vertex set $V(J) = V(H) \cup Z \cup W \cup A' \cup D'$, where Z, W, A', D' are disjoint groups of additional vertices to create extra

"flow" paths. Concretely, Z and W will be copies of original vertex set $V(G)$, and A' and D' are additional sets of vertices of size $|A'| = |D'| = 10n$. Let

$$\begin{aligned} E_Z = &\{(a,z)|a \in A, z \in Z, (a,z) \in E(G)\} \cup \\ &\{(d,z)|d \in D, z \in Z, (d,z) \in E(G)\} \end{aligned}$$

and

$$\begin{aligned} E_W = &\{(a,w)|a \in A, w \in W, (a,w) \notin E(G)\} \cup \\ &\{(d,w)|d \in D, w \in W, (d,w) \notin E(G)\} \end{aligned}$$

be the extra edges to "equalize the deviation" of all pairs (a,d) between A and D. Let

$$E_{A'} = \{(a,a') \mid a \in A, a' \in A'\}$$

and

$$E_{D'} = \{(d,d') \mid d \in D, d' \in D'\}$$

be extra edges that bring large deviations to pairs inside A or D. Finally, we construct a set of extra edges

$$E_{AD} = \{(a,d) \mid a \in A, d \in D, (a,d) \notin E_{\text{dem}}\}$$

to prevent non-demand pairs $(V(G) \times V(G)) \setminus E_{\text{dem}}$ from affecting the Steiner connectivity value. The whole edge set $E(J) = V(H) \cup E_Z \cup E_W \cup E_{A'} \cup E_{D'} \cup E_{AD}$.

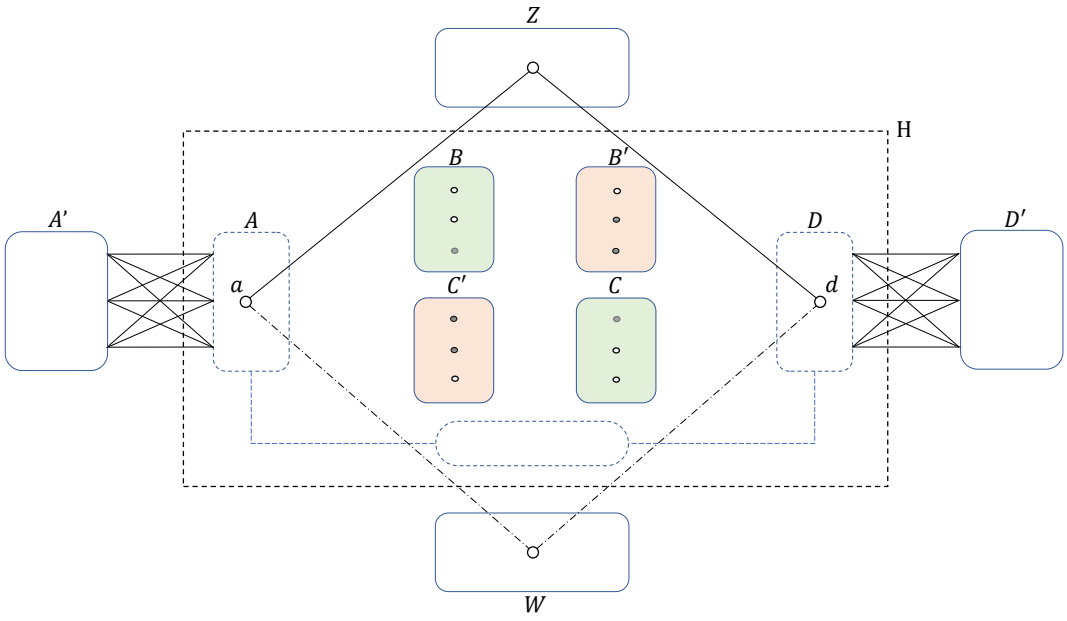

Figure 4: The graph J. There are bipartite cliques between A and A', D and D', and the edges between A, D and Z, W are linked based on E_Z, E_W.

The correctness of the reduction will be established by the following lemmas.

Lemma 4.2. *In graph J, for each $a_1, a_2 \in A$ with $a_1 \neq a_2$ and $d_1, d_2 \in D$ with $d_1 \neq d_2$, we have $\kappa_J(a_1, a_2) \geq 10n$ and $\kappa_J(d_1, d_2) \geq 10n$.*

Proof. For each pair $a_1, a_2 \in A$ with $a_1 \neq a_2$, we can construct at least $10n$ internally vertex-disjoint paths from a_1 to a_2 via vertices in A' and edges in $E_{A'}$, so $\kappa_J(a_1, a_2) \geq 10n$. Similarly, $\kappa_J(d_1, d_2) \geq 10n$.

□

Lemma 4.3. *In graph J, for each $a \in A$, $d \in D$ with $(a, d) \notin E_{\text{dem}}$, $\kappa_J(a, d) \geq 5n + 1$.*

Proof. From our construction, the number of nodes in Z adjacent to both a and d is

$$|\{z \in Z | (a, z) \in E_Z, (d, z) \in E_Z\}| = |N_G(a) \cap N_G(d)|,$$

and the number of nodes in W adjacent to both a and d is

$$|\{w \in W | (a, w) \in E_Z, (d, w) \in E_Z\}| = |\bar{N}_G(a) \cap \bar{N}_G(d)|.$$

Furthermore, there is an individual edge in E_{AD} directly connecting a and d for each such $(a, d) \notin E_{\text{dem}}$.

Therefore, we have $|N_G(a) \cap N_G(d)| + |\bar{N}_G(a) \cap \bar{N}_G(d)| + 1$ extra vertex-disjoint paths from a to d. Furthermore, Inequality (1) from Lemma 3.7 shows that $\kappa_H(a, d) \geq 4n + |N_G(a) \cap \bar{N}_G(d)| + |\bar{N}_G(a) \cap N_G(d)|$, so we can construct a vertex-disjoint path set with size at least

$$\begin{aligned} &4n + |N_G(a) \cap \bar{N}_G(d)| + |\bar{N}_G(a) \cap N_G(d)| \\ &\quad + |N_G(a) \cap N_G(d)| + |\bar{N}_G(a) \cap \bar{N}_G(d)| + 1 = 5n + 1, \end{aligned}$$

which implies $\kappa_J(a, d) \geq 5n + 1$.

□

Lemma 4.4. *In graph J, for each $a \in A$, $d \in D$ with $(a, d) \in E_{\text{dem}}$, there is a 4-clique containing a and d in G if and only if $\kappa_J(a, d) \geq 5n + 1$.*

Proof. First, assume there is a 4-clique containing a and d in G. From Lemma 3.7, we can construct $4n + |N_G(a) \cap \bar{N}_G(d)| + |\bar{N}_G(a) \cap N_G(d)| + 1$ internally vertex-disjoint paths connecting a and d in H. An argument similar to the proof of Lemma 4.3 shows that we have extra $|N_G(a) \cap N_G(d)| + |\bar{N}_G(a) \cap \bar{N}_G(d)|$ internally vertex-disjoint paths via vertices in Z, W and edges in E_Z, E_W. Therefore, in graph J, we can find a set of internally vertex-disjoint paths from a to d with size

$$\begin{aligned} &4n + |N_G(d) \cap \bar{N}_G(a)| + |N_G(a) \cap \bar{N}_G(d)| + 1 \\ &\quad + |N_G(a) \cap N_G(d)| + |\bar{N}_G(a) \cap \bar{N}_G(d)| = 5n + 1. \end{aligned}$$

Now assume there is no 4-clique containing a and d in G. From Lemma 3.2, we define $J_{ad} = J \setminus ((A \cup D) \setminus \{a, d\})$ by removing vertices in A and D other than a and d, and then we have

$$\kappa_J(a, d) = \kappa_{J_{ad}}(a, d) + |A| + |D|. \tag{8}$$

Note that although the source-sink isolating gadget $Q(A, D)$ is constructed by applying Lemma 3.2 on graph H and vertex sets A, D, the construction of $Q(A, D)$ is independent from H, so we can also apply Lemma 3.2 on graph J and vertex sets A, D.

Next, we will show

$$\kappa_{J_{ad}}(a,d) \leq \kappa_{H_{ad}}(a,d) + |N_G(a) \cap N_G(d)| + |\bar{N}_G(a) \cap \bar{N}_G(d)|, \tag{9}$$

where $H_{ad} = H \setminus ((A \cup D) \setminus \{a, d\})$ as defined in the proof of Lemma 3.7. The reason is that, compared to H_{ad}, the extra vertices in J_{ad} are in A', D', Z, W, which are only directly connected to a or d. Let C_H be the minimum vertex cut of size $|C_{H_{ad}}| = \kappa_{H_{ad}}(a,d)$ whose removal disconnects a and d in H_{ad}. Let

$$\begin{aligned} C_{J_{ad}} =& C_{H_{ad}} \cup \{z \in Z \mid (a,z) \in E_Z, (d,z) \in E_Z\} \cup \\ & \{w \in W \mid (a,w) \in E_Z, (d,w) \in E_Z\}. \end{aligned}$$

We can easily observe that removing $C_{J_{ad}}$ will disconnect a and d in J_{ad}, which immediately implies Inequality (9).

Finally, combining Equation (8) and Inequality (9), we have

$$\begin{aligned} \kappa_J(a,d) \leq& \kappa_{H_{ad}}(a,d) + |A| + |D| + |N_G(a) \cap N_G(d)| \\ & + |\bar{N}_G(a) \cap \bar{N}_G(d)|. \end{aligned}$$

From Lemma 3.7, we know that when there is no 4-clique containing a and d in G, $\kappa_H(a,d) = 4n + |N_G(a) \cap \bar{N}_G(d)| + |\bar{N}_G(a) \cap N_G(d)|$. Combining $\kappa_H(a,d) = \kappa_{H_{ad}}(a,d) + |A| + |D|$ (by Lemma 3.2 again),

$$\begin{aligned} \kappa_J(a,d) \leq& \kappa_{H_{ad}}(a,d) + |A| + |D| \\ & + |N_G(a) \cap N_G(d)| + |\bar{N}_G(a) \cap \bar{N}_G(d)| \\ =& \kappa_H(a,d) + |N_G(a) \cap N_G(d)| + |\bar{N}_G(a) \cap \bar{N}_G(d)| \\ =& 4n + |N_G(a) \cap \bar{N}_G(d)| + |\bar{N}_G(a) \cap N_G(d)| \\ & + |N_G(a) \cap N_G(d)| + |\bar{N}_G(a) \cap \bar{N}_G(d)| \\ =& 5n < 5n + 1. \end{aligned}$$

□

We are now ready to prove Theorem 4.1.

Proof of Theorem 4.1. Given an edge-universal 4-clique instance G with a set E_{dem} of demand edges. We first construct the Steiner vertex connectivity instance J with terminal set $A \cup D$ as shown above. Then G with E_{dem} is a yes instance of the edge-universal 4-clique problem if and only if the Steiner vertex connectivity of $A \cup D$ in J is at least $5n + 1$, by Lemmas 4.2 to 4.4.

For time analysis, note that $V(J) = O(n)$ by construction. Therefore, an $O(n^{4-\epsilon})$-time combinatorial algorithm for the Steiner vertex connectivity problem will imply an $O(n^{4-\epsilon})$ time combinatorial algorithm for the edge-universal 4-clique problem, which contradicts Conjecture 2.3.

□

5 The Upper Bounds

In this section, we will show the upper bounds of the APVC problem and SSVC problem in undirected unweighted sparse graphs (see Theorem 5.1 and Theorem 5.2 respectively). We only prove Theorem 5.1 in detail and briefly mention the proof of Theorem 5.2 since they are analogous.

Theorem 5.1. *Given an undirected unweighted graph G with n vertices and m edges, there is a randomized algorithm that solves the APVC problem in $\hat{O}(m^{\frac{11}{5}})$ time with high probability. Furthermore, assuming Conjecture 2.5, the running time of this algorithm becomes $\hat{O}(m^2)$.*

Theorem 5.2. *Given an undirected unweighted graph G with n vertices, m edges and a source vertex, there is a randomized algorithm to solve the SSVC problem in $\hat{O}(m^{\frac{5}{3}})$ time with high probability. Furthermore, assuming Conjecture 2.5, the running time becomes $\hat{O}(m^{1.5})$.*

A key subroutine to obtain the algorithm in Theorem 5.1 is the subroutine shown in Lemma 5.3, which can capture all vertex connectivity bounded by k. The proof of Lemma 5.3 has been shown in [IN12, PSY22] and we defer it to Section 5.1 for completeness.

Lemma 5.3. *Given an n-vertex m-edge undirected unweighted graph G and a threshold k, there is a randomized algorithm that computes the value of $\min\{\kappa_G(u,v), k\}$ for all vertex pairs (u,v) in $\hat{O}(mk^3+n^2)$ time with high probability. Furthermore, assuming Conjecture 2.5, the running time of this algorithm becomes $\hat{O}(mk^2+n^2)$.*

We now complete the proof of Theorem 5.1 using Lemma 5.3.

Proof of Theorem 5.1. As discussed in Section 1.2, our APVC algorithm will handle vertex connectivity between high-degree vertices and vertex connectivity involving low-degree vertices separately. Let k be a degree threshold which will be chosen later. Given an input graph G, let $V_h = \{v \in V(G) \mid \deg_G(v) > k\}$ and $V_l = \{v \in V(G) \mid \deg_G(v) \leq k\}$.

Note that for each vertex pair (u,v) such that $u \in V_l$ or $v \in V_l$, the vertex connectivity $\kappa_G(u,v)$ is at most k. By Lemma 5.3, the vertex connectivity of all such pairs can be computed exactly in time $\hat{O}(mk^3+n^2)$. Now consider all remaining pairs (u,v) with $u \in V_h$ and $v \in V_h$. Since $|V_h| \leq O(m/k)$ by definition, there are $O(m^2/k^2)$ remaining pairs. The vertex connectivity of each pair can be computed in $\hat{O}(m)$ time using one max flow call [CKL+22], so totally $\hat{O}(m^3/k^2)$ time suffices for remaining pairs.

The total running time of the above algorithm is then $\hat{O}(mk^3+n^2+m^3/k^2)$, by choosing $k = \Theta(m^{2/5})$, the running time will be $\hat{O}(m^{11/5})$. Assuming Conjecture 2.5, the running time will be improved to $\hat{O}(mk^2+n^2+m^3/k^2)$, which is $\hat{O}(m^2)$ by choosing $k = \Theta(m^{1/2})$. □

The proof of Theorem 5.2 is analogous and we sketch it below.

Proof of Theorem 5.2. Let $s \in V(G)$ be the source. Similarly, we partition $V(G)$ into two set $V_h = \{v \in V(G) \mid \deg_G(v) > k\}$ and $V_l = \{v \in V(G) \mid \deg_G(v) \leq k\}$ where k is the degree threshold. The vertex connectivity $\kappa_G(s,v)$ for all pairs (s,v) such that $v \in V_l$ can be computed in $\hat{O}(mk^2+n)$ time, using a subroutine analogous to Lemma 5.3. The vertex connectivity of remaining pairs can be computed in $\hat{O}(m^2/k)$ by trivially calling max flows.

The total running time is $\hat{O}(mk^2+n+m^2/k)$, which will be $\hat{O}(m^{5/3})$ by choosing $k = \Theta(m^{1/3})$. Assuming Conjecture 2.5, the running time is $\hat{O}(mk+n+m^2/k)$, which will be $\hat{O}(m^{3/2})$ by choosing $k = \Theta(m^{1/2})$.

□

5.1 Proof of Lemma 5.3

The algorithm and analysis follow the ideas in [IN12].

The algorithm is as follows. First, we create $t = O(k^2 \log n)$ sample sets $U_1, ..., U_t$, each of which is generated by sampling each vertex in $V(G)$ independently with probability $1/k$. Moreover, for each set U_i, we construct a k-Gomory-Hu tree for element connectivity using Theorem 2.4. From Theorem 2.4, for each set U_i and each pair $u, v \in U_i$, we can query $a_i(u, v) = \min\{\kappa'_{G,U_i}(u, v), k\}$ in nearly constant time. Then for each $u, v \in V(G)$, we let the final output be $a(u, v) = \min\{a_i(u, v) \mid u, v \in U_i\}$.

We then analyze the running time. The time to construct all k-Gomory-Hu trees is $\hat{O}(t \cdot mk) = \hat{O}(mk^3)$. To compute all $a_i(u, v)$ and final output $a(u, v)$, each set U_i will have size $\tilde{O}(n/k)$ w.h.p. by Chernoff bound, so the time is $\tilde{O}(t \cdot (n/t)^2) = \tilde{O}(n^2)$ (by aborting the algorithm when some U_i has size not bounded by $\tilde{O}(n/k)$). Therefore, the total running time is $\hat{O}(mk^3 + n^2)$.

The correctness is shown as follows. Consider a fixed $u, v \in V(G)$. First, $a(u, v)$ is well-defined with high probability, because for one sample set U_i, u and v are inside U_i and $a_i(u, v)$ is well-defined with probability $1/k^2$ and there are $O(k^2 \log n)$ sample sets. Given that $a(u, v)$ is well-defined, we know $a(u, v) \geq \min\{\kappa_G(u, v), k\}$ since $\kappa'_{G,U_i}(u, v) \geq \kappa_G(u, v)$ for all U_i by the definition of element connectivity. By the same reason, if $\kappa_G(u, v) \geq k$, we must have $a(u, v) = k = \min\{\kappa_G(u, v), k\}$ which is the correct answer. From now we suppose $\kappa_G(u, v) < k$ and we are going to show there exists U_i such that $u, v \in U_i$ and $a_i(u, v) = \kappa_G(u, v)$ with high probability. Note that $\kappa'_{G,U_i}(u, v) = \kappa$ if U_i is disjoint with some minimum u-v vertex cut $C_{u,v}$. From our sampling strategy, U_i contains u, v and is disjoint with $C_{u,v}$ with probability $\frac{1}{k^2}(1 - \frac{1}{k})^k = \Omega(1/k^2)$. Because there are $O(k^2 \log n)$ sample set, U_i exists with high probability.

If we assuming Conjecture 2.5, we simply subsitute k-Gomory-Hu tree for element connectivity with element connectivity Gomory-Hu tree and follow the same algorithm and analysis. We will obtain an algorithm with running time $\hat{O}(mk^2 + n^2)$.

References

[ABHS22] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Scheduling lower bounds via AND subset sum. *J. Comput. Syst. Sci.*, 127:29–40, 2022. 7

[AGI+19] Amir Abboud, Loukas Georgiadis, Giuseppe F. Italiano, Robert Krauthgamer, Nikos Parotsidis, Ohad Trabelsi, Przemyslaw Uznanski, and Daniel Wolleb-Graf. Faster algorithms for all-pairs bounded min-cuts. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 1, 4

[AKL+22] Amir Abboud, Robert Krauthgamer, Jason Li, Debmalya Panigrahi, Thatchaphol Saranurak, and Ohad Trabelsi. Breaking the cubic barrier for all-pairs max-flow: Gomory-hu tree in nearly quadratic time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 884–895. IEEE, 2022. 1, 3, 7

[AKT20] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. New algorithms and lower bounds for all-pairs max-flow in undirected graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 48–61. SIAM, 2020. 1, 3

[AKT21a] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. APMF < apsp? gomory-hu tree for unweighted graphs in almost-quadratic time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1135–1146. IEEE, 2021. 1

[AKT21b] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Subcubic algorithms for gomory-hu tree in unweighted graphs. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1725–1737. ACM, 2021. 1

[AWW16] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391. SIAM, 2016. 7

[Ben95] András A. Benczúr. Counterexamples for directed and node capacitated cut-trees. *SIAM J. Comput.*, 24(3):505–510, 1995. 7

[BGL17] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 307–318. IEEE Computer Society, 2017. 2, 6

[CKL+22] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 612–623. IEEE, 2022. 1, 19

[CKM+11] Paul F. Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 273–282. ACM, 2011. 1

[CRX15] Chandra Chekuri, Thapanapong Rukkanchanunt, and Chao Xu. On element-connectivity preserving graph simplification. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2015. 7

[DW22] Mina Dalirrooyfard and Virginia Vassilevska Williams. Induced cycles and paths are harder than you think. In *63rd IEEE Annual Symposium on Foundations of Computer*

Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022, pages 531–542. IEEE, 2022. 2, 14

[FF56] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. 1

[GH61] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961. 1, 3, 7

[GIKW19] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. *ACM Trans. Algorithms*, 15(2):23:1–23:35, 2019. 7

[GR98] Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998. 1

[HRG00] Monika Rauch Henzinger, Satish Rao, and Harold N. Gabow. Computing vertex connectivity: New bounds from old techniques. *J. Algorithms*, 34(2):222–250, 2000. 1

[IN12] Rani Izsak and Zeev Nutov. A note on labeling schemes for graph connectivity. *Inf. Process. Lett.*, 112(1-2):39–43, 2012. 19, 20

[Kar00] David R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, 2000. 1, 3

[Kle69] D. Kleitman. Methods for investigating connectivity of large graphs. *IEEE Transactions on Circuit Theory*, 16(2):232–233, 1969. 1

[KLOS14] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 217–226. SIAM, 2014. 1

[LLW88] Nathan Linial, László Lovász, and Avi Wigderson. Rubber bands, convex embeddings and graph connectivity. *Comb.*, 8(1):91–102, 1988. 4

[LNP+21] Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 317–329. ACM, 2021. 1, 3

[LP20] Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 85–92. IEEE, 2020. 1, 3

[LP21] Jason Li and Debmalya Panigrahi. Approximate gomory-hu tree is faster than n - 1 max-flows. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1738–1748. ACM, 2021. 1, 7

[LPS21] Jason Li, Debmalya Panigrahi, and Thatchaphol Saranurak. A nearly optimal all-pairs min-cuts algorithm in simple graphs. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1124–1134. IEEE, 2021. 1

[Mad16] Aleksander Madry. Computing maximum flow with augmenting electrical flows. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 593–602. IEEE Computer Society, 2016. 1

[PSY22] Seth Pettie, Thatchaphol Saranurak, and Longhui Yin. Optimal vertex connectivity oracles. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 151–161. ACM, 2022. 4, 5, 7, 19

[She13] Jonah Sherman. Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 263–269. IEEE Computer Society, 2013. 1

[Vas09] Virginia Vassilevska. Efficient algorithms for clique problems. *Inf. Process. Lett.*, 109(4):254–257, 2009. 6

[vdBLN+20] Jan van den Brand, Yin Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 919–930. IEEE, 2020. 1

[WW18] Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. 2