Faster Algorithms for Rooted Connectivity in Directed Graphs

Chandra Chekuri
University of Illinois at Urbana-Champaign, IL, USA

Kent Quanrud
Purdue University, West Lafayette, IN, USA

Abstract

We consider the fundamental problems of determining the rooted and global edge and vertex connectivities (and computing the corresponding cuts) in *directed* graphs. For rooted (and hence also global) edge connectivity with small integer capacities we give a new randomized Monte Carlo algorithm that runs in time $\tilde{O}\left(n^2\right)$. For rooted edge connectivity this is the first algorithm to improve on the $\Omega(n^3)$ time bound in the dense-graph high-connectivity regime. Our result relies on a simple combination of sampling coupled with sparsification that appears new, and could lead to further tradeoffs for directed graph connectivity problems.

We extend the edge connectivity ideas to rooted and global vertex connectivity in directed graphs. We obtain a $(1+\epsilon)$-approximation for rooted vertex connectivity in $\tilde{O}(nW/\epsilon)$ time where W is the total vertex weight (assuming integral vertex weights); in particular this yields an $\tilde{O}\left(n^2/\epsilon\right)$ time randomized algorithm for unweighted graphs. This translates to a $\tilde{O}(\kappa nW)$ time exact algorithm where κ is the rooted connectivity. We build on this to obtain similar bounds for global vertex connectivity.

Our results complement the known results for these problems in the low connectivity regime due to work of Gabow [8] for edge connectivity from 1991, and the very recent work of Nanongkai et al. [23] and Forster et al. [6] for vertex connectivity.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases rooted connectivity, directed graph, fast algorithm, sparsification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.49

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2104.07205`

Funding *Chandra Chekuri*: Supported in part by NSF grants CCF-1910149 and CCF-1907937.

Acknowledgements We thank the reviewers for their helpful comments.

1 Introduction

Let $G = (V, E)$ be a simple directed graph; that is, a directed graph with no parallel edges. Recall that G is *strongly connected* if there is a path from any vertex $a \in V$ to any vertex $b \in V$. The *edge connectivity* is the minimum number of edges that need to be removed so that G is *not* strongly connected. The corresponding set of edges is called the *minimum edge cut*. The *vertex connectivity* is the minimum number of vertices that need to be removed so that the remaining graph is not strongly connected or has only one vertex. The corresponding set of vertices is called the *minimum vertex cut*. These problems generalize to weighted settings where the edges and vertices are assigned positive weights and the goal is to find the minimum weight edge or vertex cut. Determining the edge and vertex connectivities and finding the corresponding minimum cuts are among the basic problems in graph algorithms.

This work obtains faster randomized algorithms for finding minimum edge and vertex cuts in directed graphs, especially in the dense setting. The algorithms are based on a simple technique which could be of independent interest.

Our interest is actually in the more general *rooted connectivity* problems. Let $r \in V$ be a fixed vertex, called the *root*. The *r-rooted edge connectivity* is the minimum number of edges that have to be removed so that there is some vertex in $V - r$ that r cannot reach. An algorithm for rooted edge connectivity easily implies an algorithm for edge connectivity, by fixing any root and returning the minimum of the rooted connectivity in G and the rooted connectivity in the graph obtained by reversing all the edges in G. Another important motivation for investigating rooted connectivity is the fundamental result by Edmonds [4] that the r-rooted edge connectivity equals the maximum number of edge-disjoint arboresences rooted at r. We refer the reader to [27, 7] for further connections in combinatorial optimization. Similarly, the *r-rooted vertex connectivity* is the minimum number of vertices (excluding r) that have to be removed so that r cannot reach some vertex in the residual graph. Algorithms for rooted vertex connectivity also lead to global vertex connectivity by a similar but somewhat more involved reduction. There is a long and rich history of developing algorithms for determining the edge and vertex connectivity. We first note that all of these connectivity and cut problems reduce to a polynomial number of (s, t)-cut and flow problems by standard reductions. Beyond (s, t)-flow, an interesting algorithmic landscape emerges with different running times depending on whether we are interested in edge or vertex cuts, directed or undirected graphs, and weighted or unweighted graphs.

Rooted and global edge-connectivity. We first discuss edge connectivity in directed graphs. Let λ denote either the rooted or global edge connectivity of the graph depending on the context. One can compute both via $O(n)$ (s, t)-minimum cut computations. For the simple and unweighted directed graph setting, Mansour and Schieber [21] improved on this and gave algorithms that run in $O(mn)$ time or in $O\big(\lambda^2 n^2\big)$ time for global connectivity. It was also observed by Alon (cf. [21]) that this approach can be parameterized by the minimum out-degree δ^+ to obtain a $O(n \log(\delta^+) \operatorname{EC}(m, n)/\delta^+)$ algorithm, where $\operatorname{EC}(m, n)$ denotes the running time for (s, t)-edge connectivity[1]. Gabow [8] then gave a $O\big(m\lambda \log\big(n^2/m\big)\big)$ for rooted connectivity in graphs with integer capacities. Gabow's algorithm is based on Edmonds' theorem described above. Gabow's algorithm is nearly linear time for sparse unweighted graphs, and remains the fastest algorithm for small λ for both rooted and global edge connectivity. It is interesting that Gabow's algorithm is not based on (s, t)-flow. For directed graphs with arbitrary edge capacities, Hao and Orlin [12] gave an $O\big(mn \log\big(n^2/m\big)\big)$ algorithm for rooted connectivity by adapting the push-relabel max flow algorithm; in fact their algorithm computes the (r, v)-connectivity for all $v \in V - r$. Thereafter there have been no direct running time improvements to rooted or global edge connectivity in directed graphs but we point out that there have been numerous breakthroughs in the running times for (s, t)-flow and connectivity [11, 24, 15, 19, 20, 18, 14, 28, 10]. In particular, starting with the work of Goldberg and Rao [11], the running time for (s, t)-flow is $o(mn)$ which breaks the flow-decomposition barrier. Motivated by these developments and several others, there has been a resurgence of interest and literature on faster graph algorithms for several fundamental problems. Despite these developments there has been no algorithm for rooted

[1] Depending on the context, we let $\operatorname{EC}(m, n)$ denote the running time for (s, t)-cut in either a simple or a weighted directed graph with m edges and n vertices.

Table 1 Running times for finding the minimum cut in unweighted directed graphs (i.e., $U = 1$). $\mathrm{EC}(m, n)$ denotes the running time of computing (s, t)-connectivity (in unweighted graphs). See also [27, §15.3a].

	$O(n\,\mathrm{EC}(m,n))$	Trivial. Also holds for rooted connectivity.
	$O(n\,\mathrm{EC}_\lambda(m,n))$	Matula [22]. Also holds for rooted connectivity.
	$O(mn)$, $O\big(\lambda^2 n^2\big)$	Mansour and Schieber [21]
*	$O\Big(\frac{n\log\delta}{\delta}\,\mathrm{EC}(m,n)\Big)$	Alon (cf. Mansour and Schieber [21]). δ is the minimum out-degree in the graph.
*	$O\big(m\lambda\log\big(n^2/m\big)\big)$	Gabow [8]. Also holds for rooted connectivity.
*	$\tilde{O}\big(n^2\big)$	Theorem 1. Randomized. Also holds for rooted connectivity.

edge-connectivity in simple directed graphs that is faster than $O(n^3)$ in the worst case. In this paper we obtain a nearly quadratic time algorithm which also applies to graphs with small integer capacities.[2]

▶ **Theorem 1.** *Let $G = (V, E)$ be a simple directed graph with m edges n, vertices, and integer edge weights $w : E \to [U]$. Then the minimum rooted r-cut can be computed with high probability in $\tilde{O}\big(n^2U\big)$ randomized time.*

This running time is particularly compelling when the rooted edge connectivity λ is high.

Rooted and global vertex-connectivity. We now consider (rooted) vertex connectivity in directed graphs. It is well known that for fast algorithms, global vertex connectivity is more involved than edge connectivity and the running times are more varied. While the rooted vertex connectivity can be reduced to computing $O(n)$ (s, t)-cuts, the global version, if done naively, would require $\Omega(n^2)$ calls to the (s, t)-cut problem since it is not obvious how to find a vertex that is not part of the minimum global vertex cut. There is a large body of literature and we highlight the leading (randomized) running times, where we state running times for randomized algorithms with high probability of success. Let κ denote the weight minimum vertex cut, where we assume the minimum weight of any vertex is 1. For large κ and general capacities, there is a randomized algorithm by Henzinger et al. [13] (extending the directed edge connectivity algorithm of [12]) that runs in $O(mn\log(n))$ time. For small values of κ in the unweighted setting, recent randomized algorithms by Forster et al. [6] based on *local connectivity* have obtained $\tilde{O}\big(m\kappa^2\big)$ and $\tilde{O}\big(n\kappa^3 + \kappa^{3/2}\sqrt{mn}n\big)$ running times. For more intermediate values of κ, there are also randomized $\tilde{O}\big(\kappa m^{2/3}n\big)$ and $\tilde{O}\big(\kappa m^{4/3}\big)$ time algorithms [23] as well as an $O(n^\omega + n\kappa^\omega)$ time algorithm [3], where $\omega \approx 2.3728596$ is the current exponent for fast matrix multiplication [1]. There is also recent interest in obtaining fast $(1 + \epsilon)$-approximation algorithms for minimum vertex cut [23, 6]. In particular [23] obtains a $O\big(n^\omega/\epsilon^2 + m\min\{\kappa, \sqrt{n}\}\big)$ running time (where ω denotes the exponent for matrix multiplication) and [6] obtains a randomized algorithm with running time $\tilde{O}(m\kappa/\epsilon)$. We obtain the following theorem.

[2] Here and throughout $\tilde{O}(\cdots)$ hides polylogarithmic factors in m and n. We note that the ideas introduced in this work are simple and the logarithmic factors they generate are easy to account for. However Theorem 1 also uses the recent (s, t)-flow algorithm of [28] with running time $\mathrm{EC}(m, n) = \tilde{O}\big(m + n^{1.5}\big)$, which has large logarithmic factors.

Table 2 A table of running times for finding the minimum vertex cut in unweighted directed graphs (i.e., $W = n$). $\mathrm{VC}(m, n)$ denotes the running time of computing (s, t)-vertex connectivity, and is at most $\tilde{O}\left(m + n^{1.5}\right)$ [29]. All randomized algorithms above are correct with high probability. See also [27, §15.2a] and [6].

$O\left(n^2\,\mathrm{VC}(m,n)\right)$	Trivial.
$O(n\,\mathrm{VC}(m,n)\log(n))$	Trivial. Randomized. $\kappa \leq .999n$
$O(\kappa n\,\mathrm{VC}(m,n))$	Podderyugin [25], Even and Tarjan [5]
$O(n^\omega + n\kappa^\omega)$	Cheriyan and Reif [3].
$O(\kappa mn)$, $O\left((\kappa^3+n)m\right)$	Henzinger et al. [13].
$O(mn\log(n))$	Henzinger et al. [13]. Randomized.
$O\left(\min\left\{\kappa^{5/2}, \kappa n^{3/4}\right\}m + mn\right)$	Gabow [9].
$\tilde{O}\left(\kappa m^{2/3}n\right)$, $\tilde{O}\left(\kappa m^{4/3}\right)$	Nanongkai et al. [23]. Randomized.
$\tilde{O}\left(m\kappa^2\right)$, $\tilde{O}\left(n\kappa^3 + \kappa^{3/2}m^{1/2}n\right)$	Forster et al. [6]. Randomized.
$\tilde{O}\left(n^2\kappa\right)$	Corollary 3. Randomized.

▶ **Theorem 2.** *Let $G = (V, E)$ be a directed graph with m edges, n vertices, and integer vertex weights $w : V \to \mathbb{N}$. Let $r \in V$ be a fixed root vertex. Let κ be the rooted vertex connectivity from r. Let $W = \sum_{v\in V} w(v)$ be the total weight of the graph. For any $\epsilon > 0$ a $(1+\epsilon)$-approximate rooted minimum vertex cut can be computed with high probability in $\tilde{O}(m + n(W-\kappa)/\epsilon)$ randomized time; for unit weights this is $\tilde{O}(m + n(n-\kappa)/\epsilon)$. The rooted connectivity can be computed with high probability in $\tilde{O}(m + \kappa n(W-\kappa))$ time.*

Note that $W \geq n$ in the above running times. We point out that the approximation algorithm's running time is independent of κ. This large κ regime has been challenging for previous approaches. The rooted connectivity algorithm, when combined with sampling and other ideas, gives the following theorem for global vertex connectivity. As we remarked, the reduction from global to rooted is not as clean for vertex connectivity as it is for edge connectivity.

▶ **Corollary 3.** *Let $G = (V, E)$ be a directed graph with m edges, n vertices, and integer vertex weights $w : V \to \mathbb{N}$. Let $W = \sum_{v\in V} w(v)$ be the total vertex weight of the graph. Let κ be the global vertex connectivity of G. There is a randomized algorithm that for any $\epsilon > 0$ outputs a $(1+\epsilon)$-approximate minimum vertex cut with high probability in time $\tilde{O}(nW/\epsilon)$. There is a $\tilde{O}(\kappa nW)$ time randomized algorithm that computes the (exact) minimum vertex cut with high probability. In particular, for unit weights, the running time is $\tilde{O}\left(\kappa n^2\right)$.*

1.1 Key ideas

Our algorithms are based on a simple but key idea that we briefly outline below. We focus on the edge-connectivity case since the idea for vertex connectivity is essentially the same with some modifications. We would like to take advantage of recent developments on fast algorithms for (s, t)-cut and reduce to solving a small number of such cut problems in a black box fashion (unlike the approach of [12] based on the properties of a specific flow

algorithm). For undirected graph global connectivity there has been very recent exciting progress by Li and Panigrahi [17] reducing to a *logarithmic* number of (s,t)-cuts. However, the technique makes strong use of the symmetry of the edge-cut function which are absent in the directed graph setting. In a different direction the work of Nanongkai, Saranurak, and Yingchareonthawornchai [23] and follow up improvements by Forster et al. [6], developed fast algorithms for global connectivity based on *local connectivity* and *randomization*. At a high-level they use sampling to identify two vertices s,t on the opposite sides of a cut and then reduce to (s,t)-cut, or they use a local-connectivity algorithm from each vertex $v \in V$. This approach is particularly well-suited for *small* connectivity.

For directed graph edge connectivity Gabow's algorithm with running time $\tilde{O}(m\lambda)$ is very good. In order to beat $O(n^3)$ in the worst case, the bottleneck is the dense graph regime with high connectivity. We have two main ideas that are particularly well suited to this regime. First, we focus on the rooted case even though it may appear to be more difficult than the global connectivity case. The global connectivity can be much smaller than the rooted connectivity; for instance the graph may not be strongly connected, in which case the global edge connectivity is 0, while the rooted connectivity for a particular root can still be $\Omega(n)$. Consider rooted connectivity from a given vertex r. In order to reduce to (s,t)-cut we would like to find a node t such that t is the sink side of a minimum r-cut. Let $T \subseteq V$ be a sink side of a minimum r-cut and hence $\lambda = |\delta^-(T)|$; here $\delta^-(T)$ denotes the set of edges entering T. If $|T|$ is large we can randomly sample a small number of vertices and we will succeed with good probability in finding a vertex from T. Therefore the difficult case is when $|T|$ is small and this is the setting in which we make our key observation: if the graph is *simple* (or edge capacities are small) and the sink side of a minimum r-cut is small (but not a singleton!), then T cannot have a high-degree vertex. How can we take advantage of this? Since we are working with the rooted problem, we can shrink all high-degree vertices into the root r! In other words we can *sparsify* the graph if the sink side is small and compensate for the higher sampling rate (and larger number of (s,t)-cut computations) we need to find a vertex on the sink side. Simple in retrospect, this tradeoff between sparsification and sampling rate coupled with guessing the size of the sink component gives us the overall algorithm with some additional technical work. We believe that our high-level idea will find use in other contexts when combined with other techniques.

Recent related work. Several recent papers have obtained results that are relevant to this work. Li et al. [16] obtained an $\tilde{O}\big(mn^{1-1/12+o(1)}\big)$ time algorithm for vertex connectivity in directed and unweighted graphs. A followup work by one of the authors of this paper obtained $(1+\epsilon)$-approximation algorithms for *weighted* graphs, for rooted and global, edge and vertex connectivity, with $\tilde{O}\big(n^2/\epsilon^{O(1)}\big)$ running times [26]. Improved exact algorithms for weighted edge connectivity have also been obtained very recently in [2].

2 Edge connectivity

In this section, we prove the main theorem for edge connectivity, Theorem 1. To this end, we will first introduce the main key lemma, called the *Rooted Sparsification Lemma*, in Section 2.1. In Section 2.2, we give a lemma that applies the Rooted Sparsification Lemma to give a faster algorithm when the number of vertices in the sink component is known to be in a fixed interval between 1 and n. Theorem 1 is then proven in Section 2.4, applying the ideas from Section 2.2 to each of a small family of intervals.

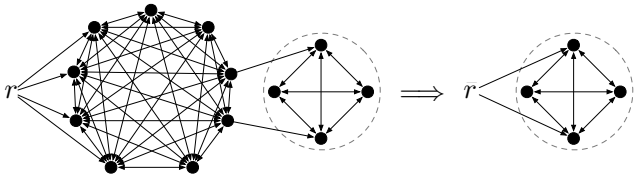


Figure 1 An example of the Rooted Sparsification Lemma in action. In particular, contracting the high in-degree vertices into r leaves the sink component of the minimum r-cut intact.

2.1 The Rooted Sparsification Lemma for Edge Connectivity

We introduce the key technical ingredient that we call the Rooted Sparsification Lemma. This lemma says that if the sink component of the minimum r-cut is small, then unless it is a singleton cut (which is easy to find directly), we can contract all vertices with high in-degree into the root while preserving the minimum rooted cut exactly. The result is a smaller and sparser graph in which we can find the minimum rooted cut faster. Later we will see that the running time saved by operating on a smaller graph makes up for the difficulty in identifying a vertex from a smaller sink component.

▶ **Lemma 4.** *Let $G = (V, E)$ be a simple directed graph with m edges, n vertices, and edge weights $w : E \to [1, U]$. Let $r \in V$ be a fixed root vertex. Let $k \in \mathbb{N}$. Consider the graph $\bar{G}$ obtained by contracting all vertices with weighted in-degree $\geq (1 + U)k$ into r. Let $\bar{r}$ denote the contracted vertex in $\bar{G}$. Then we have the following.*

1. *$\bar{G}$ is a multigraph with less than $(1 + U)nk$ edges.*
2. *If the minimum number of vertices in a sink component of a minimum r-cut has greater than 1 and less than or equal to k vertices, then the minimum r-cut and the minimum $\bar{r}$-cut are the same.*

Note that contraction cannot reduce the value of r-cut. An example illustrating the lemma is given in Figure 1. The proof is in two steps.

Small sinks make small cuts (except for singletons). The first step towards the Rooted Sparsification Lemma for edge connectivity is the following basic observation relating the connectivity to the number of vertices in the sink component of a minimum rooted cut. For simple graphs (i.e., $U = 1$), the following lemma says that *except for the case where the minimum rooted cut is achieved by a singleton*, the rooted connectivity is less than the number of vertices in the sink component of the cut. With capacities between 1 and U, we obtain a similar inequality except scaled by U. See Figure 2 for an illustration of the following lemma.

▶ **Lemma 5.** *Let $G = (V, E)$ be a simple directed graph with m edges, n vertices, and edge weights $w : E \to [1, U]$. Let $r \in V$ be a fixed root vertex. Let λ be the rooted edge connectivity from r. Let k be the minimum number of vertices in a sink component of a minimum r-cut. Then either $k = 1$ or $\lambda < Uk$.*

Proof. Let T be the set of vertices on the sink-side of a cut with λ edges. Suppose $k = |T| > 1$. Every vertex in T has weighted in-degree $> \lambda$. Consider all edges with head in T. Because G has capacities between 1 and U, of all the edges with head in T, at most $k(k-1)U$ total weight have their tail in T as well. Thus $\lambda > k\lambda - k(k-1)U$. Rearranging, we have $k(k-1)U > (k-1)\lambda$, hence $kU > \lambda$. ◀

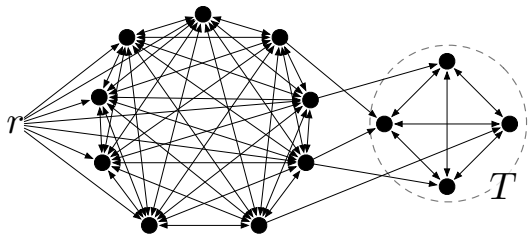


Figure 2 The set of vertices T has 4 vertices and there are 5 edges crossing into T. Lemma 5 implies that T cannot be the sink component of the minimum r-cut. Indeed, there are singleton cuts of degree 4 inside T.

▶ Remark 6. The above argument is simple and (unsurprisingly) we realized that a similar line of reasoning has appeared in previous work [21] (though towards a different algorithmic approach and not in the context of rooted connectivity).

Small sinks are sparse sinks. We now prove the Rooted Sparsification Lemma, Lemma 4. The high level argument is very simple and we first give an informal argument to emphasize the intuition. If the sink component of the minimum r-cut is small, then by Lemma 5, the minimum r-cut is also small. Suppose for the sake of discussion that the graph is simple (i.e., $U = 1$). If both the minimum r-cut and the sink component are small and the graph is simple, then every vertex in the sink component has small in-degree. The contrapositive implies that every high in-degree vertex is on the source side of the cut. Thus the high in-degree vertices can be safely contracted into the root.

Proof of the rooted sparsification lemma. Recalling the statement of the lemma, it is easy to see that contracting all vertices with weighted in-degree $\geq (1+U)k$ into r results in a multigraph $\bar{G}$ in which every vertex has weighted in-degree $< (1+U)k$, and hence there are at most $(1+U)nk$ edges total.

Let T be the sink component of a minimum r-cut. Observe that contracting into r cannot decrease the edge connectivity. If one can show that no vertices in T are contracted into $\bar{r}$, then T is the sink component of a minimum $\bar{r}$-cut as well.

By Lemma 5, the minimum r-cut has size $\lambda < Uk$. Because G is simple and T has $\leq k$ vertices, every vertex in T has in-degree less than $\lambda + k < (1+U)k$. Thus any contracted vertex is outside of T. This completes the proof. ◀

2.2 Rooted connectivity for a fixed range of component sizes

Applying the Rooted Sparsification Lemma usefully requires a fairly tight upper bound on the number of vertices in the sink component of the minimum r-cut. In this section, we assume we are given a lower bound k_1 and an upper bound k_2 on the number of vertices in the sink component, and develop algorithms for the minimum rooted cut in this parametrized regime. The running times are decreasing in k_1 and increasing in k_2; that is, they are better for tighter bounds on the number of vertices in the sink component.

▶ **Lemma 7.** *Let $G = (V, E)$ be a simple directed graph with m edges, n vertices, and edge weights $w : E \to [1, U]$. Let $r \in V$ be a fixed root vertex. Let λ be the rooted edge connectivity from r. Let $k_1, k_2 \in \mathbb{N}$ with $1 \leq k_1 \leq k_2 \leq n$. Suppose the sink component of the minimum r-cut has between k_1 and k_2 vertices. Then the minimum r-cut can be computed with constant probability in*

$$O\left(m + \frac{n}{k_1}(\mathrm{EC}(\min\{m, nk_2U\}, n))\right) \text{ time.}$$

Proof. We first consider the case $k_1 > 1$. By Lemma 4, we can reduce the number of edges to $m' = O(k_2 nU)$ while preserving the r-cut and retaining all k_1 or more vertices in the sink-side component. Let us sample $O(n/k_1)$ sink vertices t in the remaining graph uniformly at random, and compute the minimum (r,t)-cut for each. This takes $\mathrm{EC}(\min\{m, m'\}, n) = \mathrm{EC}(\min\{m, k_2 nU\}, n)$ time for each instance, as desired. With constant probability, at least one sink is sampled out of the sink component of the minimum r-cut, which will return the minimum r-cut.

If $k_1 = 1$, then we must also address the possibility of a singleton cut. We apply the above for $k_1 = 2$ and compare the output to all of the singleton r-cuts, and output the smallest of these cuts. ◀

2.3 Rooted connectivity for small sink components

▶ **Lemma 8.** *Let $G = (V, E)$ be a simple directed graph with m edges, n vertices, and integer edge weights $w : E \to [U]$. Let $r \in V$ be a fixed root vertex. Let $k \in \mathbb{N}$ be a given parameter. There is a deterministic algorithm that runs in $O\big(m + nk^2U^2 \log(\max\{n/kU\})\big)$ time and returns an r-cut with the following guarantee. If the sink component of a minimum r-cut has at most k vertices, then the algorithm will return a minimum r-cut.*

Proof. If the sink side of the minimum cut has less than k vertices, then via Lemma 5, either a singleton induces a minimum r-cut, or the minimum r-cut has size $\lambda < Uk$. For the latter case, we apply the rooted sparsification lemma and reduce the graph to $O(nkU)$ edges while preserving the minimum r-cut. We apply Gabow's algorithm [8] to the sparsified graph and it runs $O\big(nk^2U^2 \log(\max\{1, n/kU\})\big)$ time, and either finds a minimum rooted cut or certifies that the r-cut value in the sparsified graph has value $\geq kU$. We compare the output with all singleton r-cuts. ◀

2.4 Algorithm for rooted edge connectivity

We now prove the main theorem for edge connectivity, Theorem 1. By Lemma 7, if the number of vertices in the sink component is known, then we can reduce very efficiently to (s,t)-connectivity by either sparsifying the graph (if the number is small) or easily guessing a sink (if the number is large). More generally, we can pursue both strategies relative to any given upper and lower bounds on the number of vertices in the sink component. Meanwhile, for small component sizes (that are not singletons), we can still sparsify the graph, while the cut size must be small, which combine to produce fast running times via [8] in Lemma 8. The only unknown is the number of vertices in the sink component. Here we guess the number of vertices up to a constant factor, which only requires enumerating a logarithmic number of guesses. We restate Theorem 1 for the sake of convenience.

▶ **Theorem 1.** *Let $G = (V, E)$ be a simple directed graph with m edges n, vertices, and integer edge weights $w : E \to [U]$. Then the minimum rooted r-cut can be computed with high probability in $\tilde{O}\big(n^2U\big)$ randomized time.*

Proof. Let $\ell \in [n]$ be a parameter to be determined. The sink component of the minimum r-cut either (a) is a singleton, (b) has at most ℓ vertices, or (c) has between 2^i and 2^{i+1} vertices for some $i \geq \lfloor \log \ell \rfloor$. For each of these categories we apply a subroutine and take the minimum of the cut values found.

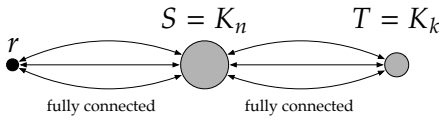
Singleton cuts are easy to evaluate in $O(m)$ time. Let $i_0 = \lfloor \log \ell \rfloor$ and $i_1 = \max\{\lceil \log m/nU \rceil, i_0 + 1\}$. For $i = i_0, \ldots, i_1 - 1$, let $k_i = 2^i$. Let $k_{i_1} = n$. For (b) we invoke Lemma 8 with maximum sink component k_{i_0}. To address (c), for each $i = i_0, \ldots, i_1 - 1$, we invoke Lemma 7 $O(\log n)$ times with lower bound k_i and upper bound k_{i+1} on the number of vertices in the sink component. We use $\mathrm{EC}(m, n) = \tilde{O}\left(m + n^{1.5}\right)$ [28]. The combined running time is $\tilde{O}\left(n^2 U + \frac{n^{2.5}}{\ell} + n\ell^2 U^2\right)$. For $\ell = \sqrt{n}/U$, this gives the claimed running time. ◀

3 Rooted and global vertex connectivity

In this section, we describe and analyze the approximation algorithms for rooted and global vertex connectivity. The high-level approach is similar to the previously discussed algorithm for edge connectivity. The first step, Lemma 10, is a variant of the Rooted Sparsification Lemma that applies to (approximate) vertex connectivity. It plays a similar role as its counterpart for edge connectivity, allowing one to sparsify the graph when the sink component of the minimum rooted vertex cut is small. The proof of Lemma 10 is given Section 3.1. We then give an algorithm specific to (roughly) the number of vertices in the sink component in Section 3.2. We use this algorithm as a subroutine in the final algorithm for approximate rooted connectivity in Section 3.4. In Section 3.5, we give the reduction from approximate global vertex connectivity to approximate rooted vertex connectivity. The exact global vertex connectivity algorithm for integer weights follows from an appropriate choice of error parameter.

3.1 Rooted sparsification for approximate vertex connectivity

Recall that a key idea in the algorithm for (rooted) edge connectivity was the Rooted Sparsification Lemma, which allows us to substantially decrease the number of edges when the sink component of the minimum rooted cut is small. Underlying the rooted sparsification lemma for edge connectivity was a direct relation between the size of the sink component and the weight of the minimum edge cut – Lemma 5 in Section 2.1. But this relation does not hold for vertex connectivity, even in unweighted and undirected graphs – even if the sink component is small, the vertex in-cut can be very large. For example, for arbitrarily large n and any fixed constant k, let $S = K_n$ be a clique of size n and let $T = K_k$ be a clique of size k. Add edges between all $s \in S$ and all $t \in T$. Let r be an additional root vertex connected to every vertex in S. Then T is the sink component of the minimum vertex r-cut. It has a constant number of vertices, k, while the size of the vertex cut, n, is arbitrarily large.



That said, we show that a useful sparsification is possible if we relax to *approximating* the rooted vertex connectivity, and qualify the lemma by the assumption that no singleton cut already represents a good approximation. To this end, let $u, v \in V$. We say that u is an *in-neighbor* of v if $(u, v) \in E$. We denote the set of in-neighbors of a vertex v by $N^-(v) \stackrel{\text{def}}{=} \{u \in V : (u, v) \in E\}$. The definition of in-neighbors naturally extends to sets of vertices; for $S \subset V$ we define $N^-(S) = (\cup_{v \in S} N^-(v)) \setminus S$. The *weighted in-degree* of v is defined as the total weight of all in-neighbors of v. Similarly we define the set of *out-neighbors* of a vertex v, denoted $N^+(v)$, as $N^+(v) \stackrel{\text{def}}{=} \{u \in V : (u, v) \in E\}$, and the weighted out-degree of v, denoted $\deg^+(v)$, as the sum of weights over $N^+(v)$.

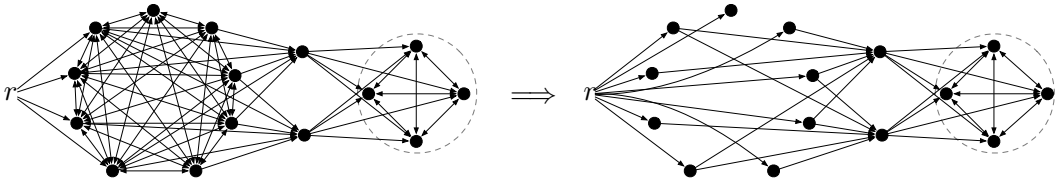


Figure 3 An example of the Rooted Sparsification Lemma for vertex connectivity in action. In the input graph on the left, minimum vertex r-cut has size 2 and the sink component (circled) has 4 vertices. The minimum in-degree (other than r) is 5. On the right hand side, all vertices with in-degree ≥ 9 have all their incoming edges replaced with a single edge from r. The minimum vertex r-cut is again 2 and the sink-component of the minimum r-cut remains unchanged.

Our first lemma gives an approximate relationship between the weight of the minimum weight rooted vertex cut and the weight of the sink component of the minimum weight rooted vertex cut.

▶ **Lemma 9.** *Let $\epsilon > 0$ be fixed. Let $G = (V, E)$ be a directed graph with m edges, n vertices, and vertex weights $w : V \to [1, \infty)$. Let $r \in V$ be a fixed root vertex. Let κ be the rooted vertex connectivity from r. Suppose the in-neighborhood of every non-root vertex has total weight greater tha $(1 + \epsilon)\kappa$. Then the minimum vertex r-cut has more than $\epsilon\kappa$ weight in the sink component.*

Proof. Let the minimum r-cut be of the form $N^-(S)$, where $S \subseteq V - r$. To prove the claim it suffices to show that $w(S) > \epsilon\kappa$.

For any vertex $v \in S$, by assumption, total weight of in-neighbors is more tha $(1 + \epsilon)\kappa$. At most κ weight of these in-neighbors are in the minimum vertex r-cut, $N^-(S)$. This implies that v has more than $\epsilon\kappa$ weight of in-neighors in S, and hence $\sum_{s \in S} w(s) > \epsilon\kappa + 1$ (where the extra 1 is for the weight of v). ◀

▶ **Lemma 10.** *Let $\epsilon > 0$ be fixed. Let $G = (V, E)$ be a directed graph with m edges, n vertices, and vertex weights $w : V \to [1, \infty)$. Let $r \in V$ be a fixed root vertex. Let κ be the rooted vertex connectivity from r. Suppose every vertex (excluding r) has weighted in-degree greater than $(1 + \epsilon)\kappa$. Let $k \in \mathbb{N}$. Consider the graph $\bar{G}$ obtained by replacing, for each vertex $v \in V$ with weighted in-degree $\geq (1 + 1/\epsilon)k$, all of the in-coming edges to v with a single edge from r to v. Then we have the following.*

1. *$\bar{G}$ has maximum weighted in-degree at most $(1 + 1/\epsilon)nk$.*
2. *$\bar{G}$ has at most $(1 + 1/\epsilon)nk$ edges.*
3. *If the sink component of a minimum vertex r-cut in G has weight $\leq k$, then the minimum vertex r-cut in G and $\bar{G}$ are the same.*

Proof. Let T be the sink component of a minimum rooted r-vertex cut, of minimum weight among such sink components. Suppose T has weight less than or equal to k. By Lemma 9, $\kappa < k/\epsilon$. Therefore any vertex v with weighted in-degree greater than $(1 + 1/\epsilon)k$ cannot be in T of the minimum rooted r-vertex cut. We claim that replacing the incoming edges to v does not decrease rooted vertex connectivities for r. As a thought experiment, suppose we make the replacement over two steps, where we first add the edge from r to v, and then remove the other incoming edges to v. The first step does not decrease vertex connectivities, and forces the rooted vertex connectivity from r to v to be $+\infty$. Removing the other incoming

edges to r does not effect the connectivity from r to v, so no other vertex connectivities from r are effected either. Over the two steps, then, we see that no rooted connectivities from r decrease.

On the other hand, since v is not in the sink of the minimum vertex r-cut, the rooted vertex connectivity of r does not change. ◀

3.2 Rooted vertex connectivity parametrized by sink component size

We now give an algorithm for rooted vertex connectivity parametrized by the weight of the vertices in the sink component of the minimum rooted cut. More precisely, we take as additional input two weights $k_1 \leq k_2$ and assume the sink component has weight between k_1 and k_2.

In the following, let $\mathrm{VC}(m, n)$ be the running time for vertex (s, t)-cut.

▶ **Lemma 11.** *Let $\epsilon > 0$ be fixed. Let $G = (V, E)$ be a directed graph with m edges, n vertices, and vertex weights $w : V \to [1, \infty)$. Let $r \in V$ be a fixed root vertex. Let κ be the rooted vertex connectivity from r. Let $W = \sum_{v \in V} w(v)$ be the total weight in the graph. Suppose every vertex (excluding r) has weighted in-degree greater than $(1 + \epsilon)\kappa$. Let $k_1, k_2 \in \mathbb{N}$ with $0 < k_1 < k_2$. Suppose also that the sink component of the minimum r-cut has between k_1 and k_2 total weight. Then the minimum r-cut can be computed with constant probability in*

$$O\left(m + \left(\frac{W - \kappa}{k_1}\right) \mathrm{VC}\left(\min\left\{m, \frac{k_2 n}{\epsilon}\right\}, n\right)\right)$$

randomized time.

Proof. By Lemma 10, in $O(m)$ time, we can reduce the number of edges to at most $O(k_2 n/\epsilon)$ without decreasing the rooted vertex connectivity. We sample vertices from $V' = V \setminus (\{r\} \cup N^+(r))$. Note that V' has weight at most $W - \deg^+(r) \leq W - \kappa$.

We sample $O\big((W - \deg^+(r))/k_1\big) \leq O((W - \kappa)/k_1)$ vertices $t \in V'$ independently in proportion to their weight. For each sampled vertex t, we compute the minimum (r, t)-vertex cut in the sparsified graph. With constant probability, one of these vertices t is in the sink component of the minimum vertex r-cut, and the minimum vertex (r, t)-cut is the minimum vertex r-cut. ◀

▶ Remark 12. The simple observation that the weight of $N^+(r)$ is at least κ is from [13].

3.3 Rooted vertex connectivity for small sink components

This section develops an approximation algorithm for rooted vertex connectivity specifically for the case where the sink component has small weight. The algorithm takes an upper bound k on the weight of the sink component, and guarantees an approximate minimum cut when there is a minimum rooted vertex cut where the sink component has weight at most k. The approach is inspired by the recent local connectivity algorithm of [6], and also integrates the rooted sparsification lemma. This algorithm is developed in two steps. The first step is a local cut algorithm that, given a vertex $t \in V$, searches for a small (r, t)-cut around t in time proportional to a given upper bound on the weight of the sink component. The second step first applies the rooted sparsification lemma, finds a vertex t in the sink component by random sampling, and runs the local cut algorithm for this choice of t.

The following lemma, which describes the local cut algorithm, is nearly identical to [6] except for two small modifications. First, we work with integral capacities, which does not change any arguments. Second is the inclusion of the root r which we want to keep on the opposite side of the local cut. The proof is included for the sake of completeness. In the following, the *in-volume* of a set of vertices T in a directed, edge-capacitated graph is the sum of weighted in-degrees over all vertices in T. Similarly the out-volume is defined as the sum of weighted out-degrees.

▶ **Lemma 13.** *Let $G = (V, E)$ be a directed graph with integral edge capacities. We assume that G is already available in memory in adjacency list format. Let $r, t \in V$, and let $\lambda, \ell > 0$ be given parameters. Then there is a randomized algorithm that runs in $O(\ell\lambda/\epsilon)$ time with the following guarantee.*

Let λ^* be the minimum capacity of all (r, t) cuts where the sink component has in-volume at most ℓ. If $\lambda^* < \lambda$, then with constant probability, the algorithm returns an (r, t)-cut of capacity at most $(1 + \epsilon)\lambda^*$.

Proof. Let T be the sink component of a minimum (r, t)-edge cut among those where the sink component has in-volume at most ℓ. We run a randomized variation of augmenting paths in the reversed graph G_{rev} where t is the source. Note that T now has out-volume at most ℓ in G_{rev}. We run the following subroutine for at most $1 + (1 + \epsilon)\lambda$ iterations, where each iteration routes one unit of flow from t to some chosen v.

Each iteration i runs DFS from t in the residual graph, until it either (a) visits r, (b) has traversed edges of total capacity at least $O(\ell/\epsilon)$, or (c) has explored all the edges reachable from t while failing to satisfy either (a) or (b). In event (a), we route one unit of flow to r. In event (b), we select one of the visited edges randomly in proportion to their capacity, and route one unit of flow to the endpoint of that edge. In either case, after routing, we update the residual graph by reverse (one unit of capacity) of each edge on the path from t to the selected sink. In event (c), we return the entire component of vertices reachable from t which induces an (r, t)-cut in the original graph. If, after $1 + (1 + \epsilon)\lambda$ iterations, we never reach event (c), then the algorithm terminates with failure.

We first argue that we return a $(1 + \epsilon)$-approximate (r, t)-cut with constant probability. We first point out that the total out-volume of T in the residual graph never increases, as we are reversing edges along edges along a path starting from t. Next, we observe that in each instance of event (b), where we randomly sample the endpoint of a visited edge as a sink, there is less than $\epsilon/2$ probability that this endpoint lies in T. This is because the graph search has traversed a total capacity of at least $O(\ell/\epsilon)$, and T has out-volume at most ℓ. That is, the out-volume of T represents at most an $(\epsilon/2)$-fraction of the searched edges.

Now, over the first $(1 + \epsilon)\lambda^*$ iterations, we expect to sample less than $\epsilon\lambda^*/2$ sinks from T. By Markov's inequality, we sample less than $\epsilon\lambda^*$ sinks from T over the first $(1 + \epsilon)\lambda^*$ iterations with probability at least $1/2$. In this event, if the algorithm did not find an (r, t)-cut within the first λ^* iterations, then we must have routed more than λ^* units of flow out of T – a contradiction. Thus the algorithm finds an (r, t)-cut within $(1 + \epsilon)\lambda^*$ iterations with probability at least $1/2$. Since this cut was obtained as the reachable set of t after routing at most $(1 + \epsilon)\lambda^*$ units of flow, the cut has capacity $\leq (1 + \epsilon)\lambda^*$.

It remains to prove the running time. Each iteration takes $O(\ell/\epsilon)$ time to traverse at most $O(\ell/\epsilon)$ edges. The algorithm runs for at most $O(\lambda)$ iterations. ◀

The next lemma presents the approximate rooted vertex cut algorithm that uses Lemma 13 as a subroutine. It also uses the rooted sparsification lemma to reduce the size of the graph and give stronger bounds on the volume of the sink component of the desired vertex cut.

▶ **Lemma 14.** *Let $\epsilon > 0$ be fixed. Let $G = (V, E)$ be a directed graph with m edges, n vertices, and integer vertex weights $w : V \to \mathbb{N}$. Let $r \in V$ be a fixed root vertex. Let $k \in \mathbb{N}$ and suppose that the sink component of the minimum r-cut has weight $\leq k$. Then a $(1+\epsilon)$-approximate minimum r-cut can be computed with high probability in $O\big(m + (W-\kappa)k^2 \log(n) \log(k)/\epsilon^3\big)$ randomized time.*

Proof. By Lemma 9, either a $(1+\epsilon)$-approximate minimum cut is induced by a singleton, or the minimum r-vertex cut has weight at most $O(k/\epsilon)$. The former is addressed by inspecting all singleton cuts. For the rest of the proof, let us assume the latter. By Lemma 10, we can sparsify the graph to have maximum weighted in-degree $O(k/\epsilon)$, hence at most $O(nk/\epsilon)$ total edges.

Let T be the sink component of the minimum r-cut, which has total vertex weight at most $O(k)$, and induces an r-cut with capacity $\kappa \leq O(k/\epsilon)$. Recall the standard auxiliary "split-graph" where vertex capacities are modeled as edge capacities. The high-level idea is to find a vertex $t \in T$ by random sampling and then apply Lemma 13 to the appropriate auxiliary vertices of r and t in the split graph.

To this end, we first bound the volume of the sink-component corresponding to T in the split-graph. We recall that the split graph splits each vertex v into an auxiliary "in-vertex" v^- and an auxiliary "out-vertex" v^+. For each v there is a new edge (v^-, v^+) with capacity equal to the vertex capacity of v. Each edge (u, v) is replaced with an edge (u^+, v^-) with capacity[3] equal to the vertex capacity of u. As a sink component, T maps to a vertex set T' in the split-graph consisting of (a) both copies v^- and v^+ of each vertex $v \in T$, and (b) the out-vertex v^+ of each vertex v in the vertex in-cut $N^-(T)$. For each vertex $v \in T$, v^- has (edge-)weighted in-degree equal to the vertex-weighted in-degree of v in the original graph, which is at most $O(k/\epsilon)$. This sums to $O(|T|k/\epsilon)$ over all $v \in T$. For each $v \in T$, v^+ has weighted in-degree equal to the vertex weight of T, which sums to the total vertex weight of T. Lastly, for each $v \in N^-(T)$, v^+ has weighted in-degree equal to the vertex weight of v. This sums to $\kappa \leq O(k/\epsilon)$ over all $v \in N^-(T)$. All summed up, the total in-volume of T' in the split-graph is at most $O(k/\epsilon)$ times the total vertex weight of T.

Suppose we had a constant factor estimate ℓ for the total vertex weight of T. Then we can sample $O\big((W - \deg^+(r)) \log(n)/\ell\big) \leq O((W-\kappa)\log(n)/\ell)$ vertices by weight from $V \setminus (\{r\} \cup N^+(r))$. With high probability, we sample $O(\log n)$ vertices from T. For each sampled vertex t we invoke Lemma 13 to find an (r,t)-cut, with upper bound $O(\ell k/\epsilon)$ on the volume of the sink component and $O(k/\epsilon)$ as the upper bound on the cut. With high probability, one of these calls returns a $(1+\epsilon)$-approximate cut. The total time, over all calls, would be $O\big((W-\kappa)\log(n)k^2/\epsilon^3\big)$.

Of course, we do not know the vertex weight of T *a priori*. However, we know that it is upper bounded by k, and let ℓ enumerate all powers of 2 between 1 and k. For each ℓ, run the process described above under the hypothesis that ℓ is a constant factor estimate for the total vertex weight of T. Each choice of ℓ takes $O\big((W-\kappa)\log(n)k^2/\epsilon^3\big)$ time. There are $O(\log(k))$ choices of ℓ. One of these choices of ℓ is a constant factor for the total volume of T and produces a $(1+\epsilon)$-approximate minimum (r,t)-cut with high probability. ◀

3.4 Rooted vertex connectivity

We now present the algorithm for approximate rooted vertex connectivity and prove Theorem 2. The algorithm combines the subroutine in Lemma 11 for logarithmically many ranges of weights, and Lemma 14 for sufficiently small weights. We restate Theorem 2 for the sake of convenience.

[3] Usually, this edge is set to capacity ∞, but either the weight of u or the weight of v are also valid.

▶ **Theorem 2.** *Let $G = (V, E)$ be a directed graph with m edges, n vertices, and integer vertex weights $w : V \to \mathbb{N}$. Let $r \in V$ be a fixed root vertex. Let κ be the rooted vertex connectivity from r. Let $W = \sum_{v \in V} w(v)$ be the total weight of the graph. For any $\epsilon > 0$ a $(1+\epsilon)$-approximate rooted minimum vertex cut can be computed with high probability in $\tilde{O}(m + n(W-\kappa)/\epsilon)$ randomized time; for unit weights this is $\tilde{O}(m + n(n-\kappa)/\epsilon)$. The rooted connectivity can be computed with high probability in $\tilde{O}(m + \kappa n(W-\kappa))$ time.*

Proof. Let $\kappa_0 = \epsilon\sqrt{n}$. Let $i_0 = \lfloor \log \kappa_0 \rfloor$, and let $i_1 = \max\{\lceil \log \epsilon m/n \rceil, i_0 + 1\}$ For each $i = \lfloor \log \kappa_0 \rfloor, \lfloor \log \kappa_0 \rfloor + 1, \ldots, i_1 - 1$, let $k_i = 2^i$. Let $k_{i_1} = W - \deg^+(r)$ where we recall that $\deg^+(r)$ is the weighted out-degree of r. For each i, we apply Lemma 11 with lower bound k_i and upper bound k_{i+1} on the weight of the sink component of the minimum vertex r-cut. We repeat this subroutine $O(\log n)$ times for each i to amplify the success probability from constant to high probability. We use $\mathrm{VC}(m, n) = \tilde{O}\big(m + n^{1.5}\big)$ [29]. We also apply Lemma 14 with $\epsilon\kappa_0$ has an upper bound on the sink component size. The set of all cuts obtained by these methods includes a $(1+\epsilon)$-approximate minimum r-cut with high probability, and we return the minimum of these cuts. The combined running time is

$$\tilde{O}\left(m + \frac{(W-\kappa)n}{\epsilon} + \frac{(W-\kappa)n^{1.5}}{\kappa_0} + (W-\kappa)\kappa_0^2/\epsilon^3\right) \leq \tilde{O}(m + (W-\kappa)n/\epsilon),$$

as desired. The exact bound follows by first using the approximation algorithm to obtain a constant factor estimate for κ, and then setting setting $\epsilon \leq 1/\kappa$. ◀

3.5 Global vertex connectivity

We now shift to global vertex connectivity and prove Corollary 3, which we address by reduction to the algorithm for rooted vertex connectivity above. We note that obtaining a root is slightly non-trivial because many vertices may be in the minimum weight vertex cut. We restate Corollary 3 for the sake of convenience.

▶ **Corollary 3.** *Let $G = (V, E)$ be a directed graph with m edges, n vertices, and integer vertex weights $w : V \to \mathbb{N}$. Let $W = \sum_{v \in V} w(v)$ be the total vertex weight of the graph. Let κ be the global vertex connectivity of G. There is a randomized algorithm that for any $\epsilon > 0$ outputs a $(1+\epsilon)$-approximate minimum vertex cut with high probability in time $\tilde{O}(nW/\epsilon)$. There is a $\tilde{O}(\kappa nW)$ time randomized algorithm that computes the (exact) minimum vertex cut with high probability. In particular, for unit weights, the running time is $\tilde{O}\big(\kappa n^2\big)$.*

Proof. Let κ denote the global vertex connectivity. If we sample a single vertex r in proportion to its weight, then with probability $1 - \kappa/W$, r is not in the minimum vertex cut. Then either the rooted vertex connectivity from r, or to r (i.e., from r in the graph G' with all the edges reversed), will give the rooted vertex cut. In principle we would like to apply Theorem 2 with root r in both orientations, which conditional on r not being in the minimum cut, succeeds with high probability. We amplify by repeating $L = O(\frac{W}{W-\kappa} \log n)$ times to obtain the high probability bound. Observe that the running time, via Theorem 2, is

$$\tilde{O}(mL + nW/\epsilon).$$

We would like to remove the mL factor.

To this end, observe that the m term arises from applying the rooted sparsification lemma for various estimates k of the weight of the sink component. Recall that for fixed k and ϵ, the sparsification lemma replaces, for every vertex v with in-degree $> O(k/\epsilon)$, all the incoming edges to v with a single edge from the root. Note that much of the sparsification lemma can be executed without r. In particular, we can remove all incoming edges to the high in-degree vertices without knowing r; once r is given, we add an edge from r to each of these vertices. The key point is that the first part, which takes $O(m)$ time, can be done once for all L sampled roots for each value of k. Thereafter, each of the L roots takes $O(n)$ to complete the sparsification for that root. This replaces the $\tilde{O}(mL)$ term with $\tilde{O}(nL)$, which is dominated by $\tilde{O}(nW/\epsilon)$.

For the exact algorithm, we first apply the approximation algorithm with $\epsilon = 1/2$ obtain a factor-2 approximation to κ within the claimed running time. We then apply the approximation algorithm again with $1/(2\kappa) \leq \epsilon \leq 1/\kappa$. ◀

References

1. Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021.
2. Ruoxu Cen, Jason Li, Danupon Nanongkai, Debmalya Panigrahi, and Thatchaphol Saranurak. Minimum cuts in directed graphs via $\sqrt{n}$ max-flows. *CoRR*, abs/2104.07898, 2021. `arXiv:2104.07898`.
3. Joseph Cheriyan and John H. Reif. Directed s–t numberings, rubber bands, and testing digraph k-vertex connectivity. *Comb.*, 14(4):435–451, 1994.
4. Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications (Proceedings Calgary International Conference on Combinatorial Structures and Their Applications, Calgary, Alberta, 1969; , eds.)*, pages 69–87. Gordon and Breach, New York, 1970.
5. Shimon Even and Robert Endre Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4(4):507–518, 1975.
6. Sebastian Forster, Danupon Nanongkai, Liu Yang, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Computing and testing small connectivity in near-linear time and queries via fast local cut algorithms. 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 2046–2065. SIAM, 2020.
7. András Frank. *Connections in Combinatorial Optimization*. Oxford University Press, 2011.
8. Harold N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *J. Comput. Syst. Sci.*, 50(2):259–273, 1995.
9. Harold N. Gabow. Using expander graphs to find vertex connectivity. *J. ACM*, 53(5):800–844, 2006.
10. Yu Gao, Yang P. Liu, and Richard Peng. Fully dynamic electrical flows: Sparse maxflow faster than goldberg-rao. *CoRR*, abs/2101.07233, 2021. `arXiv:2101.07233`.
11. Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998.
12. Jianxiu Hao and James B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms*, 17(3):424–446, 1994.
13. Monika Rauch Henzinger, Satish Rao, and Harold N. Gabow. Computing vertex connectivity: New bounds from old techniques. *J. Algorithms*, 34(2):222–250, 2000.
14. Tarun Kathuria, Yang P. Liu, and Aaron Sidford. Unit capacity maxflow in almost $o(m^{4/3})$ time. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 119–130. IEEE, 2020.

15 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{\text{rank}})$ iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433. IEEE Computer Society, 2014.

16 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. To appear in ACM STOC„ 2021.

17 Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In *IEEE 61st Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE Computer Society, 2020.

18 Yang P. Liu and Aaron Sidford. Faster energy maximization for faster maximum flow. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 803–814. ACM, 2020.

19 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 253–262. IEEE Computer Society, 2013.

20 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.

21 Yishay Mansour and Baruch Schieber. Finding the edge connectivity of directed graphs. *J. Algorithms*, 10(1):76–85, 1989.

22 David W. Matula. Determining edge connectivity in $o(nm)$. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 249–251. IEEE Computer Society, 1987. doi:10.1109/SFCS.1987.19.

23 Danupon Nanongkai, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Breaking quadratic time for small vertex connectivity and an approximation scheme. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 241–252, 2019.

24 James B. Orlin. Max flows in $o(mn)$ time, or better. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 765–774. ACM, 2013.

25 V. D. Podderyugin. An algorithm for finding the edge connectivity of graphs. *Vopr. Kibern.*, 2:136, 1973.

26 Kent Quanrud. Fast approximations for rooted connectivity in weighted directed graphs. *CoRR*, abs/2104.06933, 2021. arXiv:2104.06933.

27 Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

28 Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and ℓ_1-regression in nearly linear time for dense instances. *CoRR*, abs/2101.05719, 2021. arXiv:2101.05719.

29 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. *CoRR*, abs/2009.01802, 2020. arXiv:2009.01802.