Deterministic Min-cut in Poly-logarithmic Max-flows

Jason Li
Carnegie Mellon University
jasonmli02@gmail.com

Debmalya Panigrahi
Duke University
debmalya@cs.duke.edu

***Abstract*—We give a deterministic (global) min-cut algorithm for weighted undirected graphs that runs in time $O(m^{1+\epsilon})$ plus polylog(n) max-flow computations. Using the current best max-flow algorithms, this results in an overall running time of $\tilde{O}(m \cdot \min(\sqrt{m}, n^{2/3}))$ for weighted graphs, and $m^{4/3+o(1)}$ for unweighted (multi)-graphs. This is the first improvement in the running time of deterministic algorithms for the min-cut problem on general (weighted/multi) graphs since the early 1990s when a running time bound of $\tilde{O}(mn)$ was established for this problem.**

***Keywords*-minimum cut; graph algorithms.**

I. Introduction

The minimum cut of an undirected, weighted graph $G = (V, E, w)$ is a minimum weight subset of edges whose removal disconnects the graph. Finding the min-cut of a graph is one of the central problems in combinatorial optimization, dating back to the work of Gomory and Hu [1] in 1961 who gave an algorithm to compute the min-cut of an n-vertex graph using $n - 1$ max-flow computations. Since then, a large body of research has been devoted to obtaining faster algorithms for this problem. In 1992, Hao and Orlin [2] gave a clever amortization of the $n - 1$ max-flow computations to match the running time of a single max-flow computation. Using the "push-relabel" max-flow algorithm of Goldberg and Tarjan [3], they obtained an overall running time of $O(mn \log(n^2/m))$ on an n-vertex, m-edge graph. However, their amortization technique is specific to the push-label algorithm, and cannot be applied to faster max-flow algorithms that have been designed since their work. Around the same time, Nagamochi and Ibaraki [4] (see also [5]) designed an algorithm that bypasses max-flow computations altogether, a technique that was further refined by Stoer and Wagner [6] (and independently by Frank in unpublished work). This alternative method yields a running time of $O(mn + n^2 \log n)$. Prior to our work, these works yielding a running time bound of $\tilde{O}(mn)$ were the fastest *deterministic* min-cut algorithms for weighted graphs.

Starting with Karger's contraction algorithm in 1993 [7], a parallel body of work started to emerge in *randomized* algorithms for the min-cut problem. This line of work (see also Karger and Stein [8]) eventually culminated in a breakthrough paper by Karger [9] in 1996 that gave an $O(m \log^3 n)$ time *Monte Carlo* algorithm for the min-cut problem. Note that this algorithm comes to within poly-logarithmic factors of the optimal $O(m)$ running time for this problem. In this paper, Karger asks whether we can also achieve near-linear running time using a *deterministic* algorithm. Even before Karger's work, Gabow [10] showed that the min-cut can be computed in $O(m + \lambda^2 n \log(n^2/m))$ (deterministic) time, where λ is the value of the min-cut (assuming integer weights). Note that this result obtains a near-linear running time if λ is a constant, but in general, the running time can be exponential. Indeed, for general graphs, Karger's question remains open after more than 20 years. However, some exciting progress has been reported in recent years for special cases of this problem. In a recent breakthrough, Kawarabayashi and Thorup [11] gave the first near-linear time deterministic algorithm for this problem *for simple graphs*. They obtained a running time of $O(m \log^{12} n)$, which was later improved (and the algorithm considerably simplified) by Henzinger, Rao, and Wang [12] to $O(m \log^2 n \log \log^2 n)$. From a technical perspective, their work introduced the idea of using low conductance cuts to find the min-cut of the graph, a very powerful idea that we also exploit in this paper. Nevertheless, in spite of this progress, the question of designing a faster deterministic min-cut algorithm for general weighted graphs (or unweighted multi-graphs) remained open.

In this paper, we give the following result:

Theorem I.1. *Fix any constant $\epsilon > 0$. There is a deterministic min-cut algorithm for weighted undirected graphs that makes $(\lg n)^{O(1/\epsilon^4)}$ calls to s–t max-flow on a weighted undirected graph with $O(n)$ vertices and $O(m)$ edges, and runs in $O(m^{1+\epsilon})$ time outside these max-flow calls. If the original graph G is un-*

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00017

weighted, then the inputs to the max-flow calls are also unweighted.

Using the current fastest deterministic max-flow algorithms on unweighted multi-graphs (Liu and Sidford [13]) and weighted graphs (Goldberg and Rao [14]) respectively, this implies a deterministic min-cut algorithm for unweighted multi-graphs in $m^{4/3+o(1)}$ time and for weighted graphs in $\tilde{O}(m \cdot \min(\sqrt{m}, n^{2/3}))$ time.

This represents the first improvement in the running time of deterministic algorithms for the min-cut problem on general (weighted/multi) graphs since the early 1990s. Our running time is also the best known even if Las Vegas algorithms (that are more general than deterministic algorithms but more restrictive than Monte Carlo algorithms) are included. Finally, unlike the algorithm of Hao and Orlin that relied on amortizing runs of a specific max-flow algorithm, our algorithm is agnostic to the specific max-flow algorithm being used. Hence, if one were to believe the popularly held conjecture that max-flow will eventually be solved in (near-)linear time, then our algorithm will automatically yield an almost linear deterministic algorithm for the min-cut problem (assuming the max-flow algorithm is deterministic).

Roadmap. In Section II, we present the main new technical tool that we introduce in this work that we call *minimum isolating cuts*. We hope that this idea will be used for other problems in graph connectivity in the future. We then present our main result – the new deterministic min-cut algorithm – in Section III.

II. Minimum Isolating Cuts

We first introduce a few standard graph-theoretic definitions. For a graph $G = (V, E, w)$ and a subset $U \subseteq V$ of vertices, define $\partial_G U$ as the set of edges of G with exactly one endpoint in U; when the graph G is clear from context, we drop the subscript G and use ∂U instead. For a subset $F \subseteq E$ of edges, define $w(F) := \sum_{e \in F} w(e)$ as the total weight of edges in F. In particular, $w(\partial U)$ is the total weight of edges with exactly one endpoint in U.

Let us now formally define the problem we want to solve and the main theorem of this section:

Definition II.1 (Minimum isolating cuts)**.** *Consider a weighted, undirected graph $G = (V, E)$ and a subset of vertices $R \subseteq V$ ($|R| \geq 2$). The* minimum isolating cuts *for R is a collection of sets $\{S_v : v \in R\}$ such that for each vertex $v \in R$, the set S_v satisfies $S_v \cap R = \{v\}$ and has the minimum value of $w(\partial S'_v)$ over all sets S'_v satisfying $S'_v \cap R = \{v\}$.*

In other words, given a set of vertices R, the goal is to find, for every vertex v in R, a min-cut that separates v from all the other vertices in R. Our main theorem in this section, which we call the *isolating cut lemma*, gives an algorithm for finding minimum isolating cuts:

Theorem II.2. *[Isolating Cut Lemma.] Fix a subset $R \subseteq V$ ($|R| \geq 2$). There is an algorithm that computes the minimum isolating cuts for R using $\lceil \lg |R| \rceil$ calls to s–t max-flow on weighted graphs of $O(n)$ vertices and $O(m)$ edges, and takes $\tilde{O}(m)$ deterministic time outside of the max-flow calls. If the original graph G is unweighted, then the inputs to the max-flow calls are also unweighted.*

The rest of this section is devoted to proving Theorem II.2.

Order the vertices in R arbitrarily from 1 to $|R|$, and let the *label* of each $v \in R$ be its position in the ordering, a number from 1 to $|R|$ that is denoted by a unique binary string of length $\lceil \lg |R| \rceil$. Let us repeat the following procedure for each $i = 1, 2, \ldots, \lceil \lg |R| \rceil$. For each vertex v, color it red if the i'th bit of its label is 0, and blue if the i'th bit of its label is 1. Then, compute a min-cut $C_i \subseteq E$ in G between the red vertices and the blue vertices (for iteration i).

First, we show that $G \setminus \bigcup_i C_i$ partitions the set of vertices into connected components each of which contain at most one vertex of R. Let U_v be the connected component in $G \setminus \bigcup_i C_i$ containing $v \in R$. Then:

Claim II.3. *$|U_v \cap R| = \{v\}$ for all $v \in R$.*

Proof: By definition, $v \in U_v \cap R$. Suppose for contradiction that $U_v \cap R$ contains another vertex $u \neq v$. Since the binary strings assigned to u and v are distinct, they differ in their j'th bit for some j. Then, the cut C_j must separate u and v, i.e., removing the edges in C_j leaves u and v in separate components, which is a contradiction. ∎

Now, for each vertex $v \in R$, let λ_v be the minimum value of $w(\partial S)$ over all $S \subseteq V$ satisfying $S \cap R = \{v\}$, and let S^*_v be an inclusion-wise minimal set satisfying $S^*_v \cap R = \{v\}$ and $w(\partial S^*_v) = \lambda_v$. Then, we claim that the cut S^*_v does not *cross* the cut U_v, i.e.:

Claim II.4. *$U_v \supseteq S^*_v$ for all $v \in R$.*

Proof: Fix a vertex $v \in V$ and an iteration i. Let the side of the cut C_i containing v be $T^i_v \subseteq V$; we claim that $S^*_v \subseteq T^i_v$. Suppose for contradiction that $S^*_v \setminus T^i_v \neq \emptyset$. Note that $(S^*_v \cap T^i_v) \cap R = \{v\}$, which implies that:

$$w(\partial(S^*_v \cap T^i_v)) \geq \lambda_v = w(\partial S^*_v).$$

Indeed, by our choice of S_v^* to be inclusion-wise minimal, we can claim the strict inequality:

$$w(\partial(S_v^* \cap T_v^i)) > \lambda_v = w(\partial S_v^*).$$

But, by submodularity of cuts, we have:

$$w(\partial(S_v^* \cup T_v^i)) + w(\partial(S_v^* \cap T_v^i)) \leq w(\partial S_v^*) + w(\partial T_v^i).$$

Therefore, we get:

$$w(\partial(S_v^* \cup T_v^i)) < w(\partial T_v^i).$$

But $(S_v^* \cup T_v^i) \cap R = T_v^i \cap R$ since $(S_v^* \setminus T_v^i) \cap R = \emptyset$. In particular, the cut $\partial(S_v^* \cup T_v^i)$ also separates red vertices from blue vertices in the ith iteration. This contradicts the choice of $\partial T_v^i = C_i$ as the min-cut separating red vertices from blue vertices in the ith iteration.

Therefore, over all iterations i, none of the edges in the induced subgraph $G[S_v^*]$ are present in C_i. Note that $G[S_v^*]$ is a connected subgraph; therefore, it is a subgraph of the connected component U_v of $G \setminus \bigcup_i C_i$ containing v. ■

It remains to compute the desired set S_v given the property that $U_v \supseteq S_v$. Starting from G, contract $V \setminus U_v$ into a single vertex t; we want to compute the min v–t cut in the contracted graph G_v, which corresponds to a set S_v satisfying $S_v \cap R = \{v\}$ by Claim II.3. Since $\partial_{G_v} S_v^*$ is a valid v–t cut in this graph by Claim II.4, we have $w(\partial_{G_v} S_v) \leq w(\partial_{G_v} S_v^*) = w(\partial_G S_v^*) = \lambda_v$, as desired.

Note that each edge in E is either in exactly one graph G_v, or it is adjacent to t in exactly two graphs G_v. Therefore, the total number of edges over all graphs G_v is at most $2m$. We can compute the v–t min-cuts on all G_v in "parallel" through a single max-flow call on the disjoint union of all G_v. Note that if the original graph G is unweighted, then this max-flow instance is also unweighted. Finally, recovering the sets S_v and the values $w(\partial S_v)$ take time linear in the number of edges of G_v, which is $O(m)$ time over all $v \in R$.

This completes the proof of Theorem II.2. □

III. Deterministic Global Min-cut

In this section, we present our deterministic min-cut algorithm and prove our main result, Theorem I.1, which is restated below:

Theorem I.1. *Fix any constant $\epsilon > 0$. There is a deterministic min-cut algorithm for weighted undirected graphs that makes $(\lg n)^{O(1/\epsilon^4)}$ calls to s–t max-flow on a weighted undirected graph with $O(n)$ vertices and $O(m)$ edges, and runs in $O(m^{1+\epsilon})$ time outside these max-flow calls. If the original graph G is unweighted, then the inputs to the max-flow calls are also unweighted.*

Throughout the algorithm, we maintain a set $U \subseteq V$ of vertices that starts out as $U = V$ and shrinks over time. (Think of this set as the set R over which we call the isolating cut lemma.) We distinguish between the cases when U is *k-unbalanced* or *k-balanced* for some $k = \text{polylog}(n)$, as defined below.

Definition III.1 (k-unbalanced, k-balanced)**.** *For any positive integer k, a subset $U \subseteq V$ is* k-unbalanced *if there exists a side $S \subseteq V$ of some min-cut satisfying $1 \leq |S \cap U| \leq k$. More specifically, we say that U is* k-unbalanced with witness S*. The subset $U \subseteq V$ is* k-balanced *if there exists a min-cut whose two sides S_1, S_2 satisfy $|S_i \cap U| \geq k$ for both $i = 1, 2$. More specifically, we say that U is* k-balanced with witness (S_1, S_2).

We will only use this definition for subsets $U \subseteq V$ that span both sides of some min-cut, i.e., $S \cap U \neq \emptyset$ and $(V \setminus S) \cap U \neq \emptyset$ for some min-cut S. By definition, such a subset $U \subseteq V$ is either k-unbalanced or k-balanced (or possibly both, if there are multiple min-cuts in the graph). If U is k-unbalanced with witness S for some $k = \text{polylog}(n)$, then the algorithm computes a family $\mathcal{F}$ of subsets of U of size $k^{O(1)}\text{polylog}(n) = \text{polylog}(n)$ such that some subset $R \in \mathcal{F}$ satisfies $|R \cap S| = 1$. The algorithm then executes the isolating cut lemma (Theorem II.2) on each subset in $\mathcal{F}$, guaranteeing that the target set R is processed and the min-cut is found. Otherwise, U must be k-balanced with some witness (S_1, S_2). In this case, the algorithm computes a subset $U' \subseteq U$ such that $|U'| \leq |U|/2$ and both $S_1 \cap U' \neq \emptyset$ and $S_2 \cap U' \neq \emptyset$. Of course, the algorithm does not know which case actually occurs, so it executes both branches. But the second branch can only happen $O(\log n)$ times before $|U| \leq k$, at which point we can simply run s–t min-cut between all vertex pairs in U.

The algorithm is presented in Algorithm 1.

A. Unbalanced Case

In this section, we solve the case when U is k-unbalanced (line 4) for some fixed $k = \text{polylog}(n)$.

Lemma III.2 (Unbalanced case)**.** *Consider a graph $G = (V, E)$, a parameter $k \geq 1$, and a k-unbalanced set $U \subseteq V$. Then, we can compute the min-cut in $k^{O(1)}\text{polylog}(n)$ s–t max-flow computations plus $\tilde{O}(m)$ deterministic time.*

Our goal is to de-randomize the simple random process of sampling each vertex independently with

Algorithm 1 Deterministic Min-cut on $(G = (V, E))$

1: $U \leftarrow V$
2: $k \leftarrow C \log^C n$ for a sufficiently large constant $C = O(1/\epsilon^4)$
3: **while** $|U| \geq k$ **do**
4: Run Lemma III.2 on U ▷ Handles case when U is k-unbalanced (see Definition III.1)
5: Compute U' from U according to Lemma III.6 ▷ Handles case when U is k-balanced
6: Update $U \leftarrow U'$ ▷ $|U|$ shrinks by at least factor 2
7: **for** each pair of distinct $s, t \in U$ **do**
8: Compute min s–t cut in G
9: **return** smallest cut seen in lines 4 and 8

probability $1/k$. We compute a deterministic family of subsets $R \subseteq V$ such that for any subset S of size at most k (in particular, for the set witnessing the fact that U is k-unbalanced), there exists a subset R in the family with $|R \cap S| = 1$.

Lemma III.3. *For every n and $k < n$, there is a deterministic algorithm that constructs a family $\mathcal{F}$ of subsets of $[n]$ such that, for every non-empty subset $S \subseteq [n]$ of size at most k, there exists a set $T \in \mathcal{F}$ with $|S \cap T| = 1$. The family $\mathcal{F}$ has size $k^{O(1)} \log n$, every set in the family has at least two elements, and the algorithm takes $k^{O(1)} n \log n$ time.*

Before we prove Lemma III.3, we first show why it implies an algorithm for the unbalanced case as promised by Lemma III.2.

Proof of Lemma III.2: Let S be the set witnessing the fact that U is k-unbalanced. Apply Lemma III.3 with parameters $n = |U|$ and k. Map the elements of $[n]$ onto U, obtaining a family $\mathcal{F}$ of subsets of U such that for any set $S' \subseteq U$ with $|S'| \leq k$, there exists a set $R \in \mathcal{F}$ with $|R| \geq 2$ and $|R \cap S'| = 1$. In particular, for the set $S' = S \cap U$, we have $1 = |R \cap S'| = |R \cap (S \cap U)| = |R \cap S|$. Invoke Theorem II.2 on the set R to obtain, for each $v \in R$, a set S_v satisfying $S_v \cap R = \{v\}$ that minimizes $w(\partial S_v)$, along with the value $w(\partial S_v)$. Finally, output the set S_v with minimum value of $w(\partial S_v)$. To show that S_v is a min-cut of graph G, it suffices to verify that S_v is a valid cut (that is, $\emptyset \subsetneq S_v \subsetneq V$), and that $w(\partial S_v) \leq w(\partial S)$.

Since $|R| \geq 2$, the set S_v satisfies $\emptyset \subsetneq S_v \subsetneq R$, so it is a cut of the graph G. Since $|R \cap S| = 1$, for the vertex $u \in U$ with $R \cap S = \{u\}$, the set S satisfies the constraints for S_u. In particular, $w(\partial S_u) \leq w(\partial S)$. We output the set S_v minimizing $w(\partial S_v)$, so $w(\partial S_v) \leq w(\partial S_u) \leq w(\partial S)$, as promised. ∎

The rest of this section focuses on proving Lemma III.3. We first prove an easier variant, where we do not insist that every set in the family has at least two elements.

Lemma III.4. *For every n and k, there is a deterministic algorithm that constructs a family $\mathcal{F}$ of subsets of $[n]$ such that, for each subset $S \subseteq [n]$ of size at most k, there exists a set $T \in \mathcal{F}$ with $|S \cap T| = 1$. The family $\mathcal{F}$ has size $k^{O(1)} \log n$ and the algorithm takes $k^{O(1)} n \log n$ time.*

To prove Lemma III.4, we use the following derandomization building block due to [15]. The theorem below is from [16], who state it in terms of (n, k, k^2)-*splitters* (which we will not define here for simplicity).

Theorem III.5 (Theorem 5.16 from [16])**.** *For any $n, k \geq 1$, one can construct a family of functions from $[n]$ to $[k^2]$ such that for every set $S \subseteq [n]$ of size k, there exists a function f in the family whose values $f(i)$ are distinct over all $i \in S$. The family has size $k^{O(1)} \log n$ and the algorithm takes time $k^{O(1)} n \log n$.*

Proof of Lemma III.4: Apply Theorem III.5 to n and k, and for each function $f : [n] \to [k^2]$ in the constructed family, add the sets $f^{-1}(j)$ for all $j \in [k^2]$ to our family $\mathcal{F}$ of subsets of $[n]$. Fix any set $S \subseteq [n]$ of size k. For the function f guaranteed by Theorem III.5 for this set S, we have $|f^{-1}(f(i)) \cap S| = 1$ for any $i \in S$. Therefore, setting $T = f(i)$ for any $i \in S$ suffices.

This only handles subsets $S \subseteq [n]$ of size *exactly* k, but we can repeat the above construction for each positive integer $k' \leq k$. The total size and running time go up by a factor of k, which is absorbed by the $k^{O(1)}$ factors. ∎

Finally, to prove Lemma III.3, we add the condition that $\mathcal{F}$ cannot contain sets of size at most 1. Here, we will impose the additional constraint that $k < n$.

Proof of Lemma III.3: The only difference in the output is that $\mathcal{F}$ must contain no sets of size at most 1. Apply Lemma III.4 to n and k to obtain a family $\mathcal{F}_0$. Initialize a set $\mathcal{F}$ as $\mathcal{F}_0$ minus all subsets of size at most

1. For each singleton set $\{x\} \in \mathcal{F}_0$, choose k arbitrary elements in $[n] \setminus x$, and for each chosen element y, add the set $\{x, y\}$ to $\mathcal{F}$. The total size of $\mathcal{F}$ increases by at most a factor k. Now consider a subset $S \subseteq [n]$ of size at most k, and let T be a set in $\mathcal{F}_0$ with $|S \cap T| = 1$, as promised by Lemma III.4. If $|T| > 1$, then $T \in \mathcal{F}$ as well. Otherwise, if $T = \{x\}$, then since $|S \setminus x| < k$ and we chose k elements $y \in [n] \setminus x$, there exists some chosen $y \notin S$ for which $\{x, y\}$ was added to $\mathcal{F}$. This set $\{x, y\}$ satisfies $|S \cap \{x, y\}| = 1$. ∎

B. Balanced Case: Sparsifying U

If U is k-balanced, then we compute a subset $U' \subseteq U$ of size at most $|U|/2$ using *expander decompositions*, while preserving the condition that U' spans both sides of some min-cut. This section is dedicated to proving the following lemma:

Lemma III.6 (Sparsification of U)**.** *Fix any constant $\epsilon > 0$. Then, there is a constant $C = O(1/\epsilon^4)$ such that the following holds. Consider a graph $G = (V, E)$, a parameter $\phi \leq 1/(C \log^C n)$, and a set $U \subseteq V$ of vertices that is $(1 + 1/\phi)^3$-balanced with witness (S_1, S_2). Then, we can compute in deterministic $O(m^{1+\epsilon})$ time a set $U' \subseteq U$ with $|U'| \leq |U|/2$ such that $S_i \cap U' \neq \emptyset$ for both $i = 1, 2$.*

Deterministic Expander Decomposition. Our main tool will be a deterministic *expander decomposition*. We first introduce some notation. Let $G = (V, E)$ be an undirected graph with edge weights w. For disjoint vertex subsets $V_1, \ldots, V_\ell \subseteq V$, define $E(V_1, \ldots, V_\ell)$ as the set of edges $(u, v) \in E$ with $u \in V_i$ and $v \in V_j$ for some $i \neq j$. Recall that $w(F)$ is the sum of weights of edges in F; i.e., $w(E(V_1, \ldots, V_\ell))$ is the sum of weights of edges with endpoints in different vertex sets in $V_1, V_2, \ldots, V_\ell$. In particular, for a cut (A, B), we denote the edges in the cut both by $E(A, B)$ as well as the previously introduced notation ∂A (or ∂B), and the weight of the cut is correspondingly denoted $w(E(A, B))$ as well as $w(\partial A)$ (or $w(\partial B)$). For a vector $\mathbf{d} \in \mathbb{R}^V$ of entries on the vertices, define $\mathbf{d}(v)$ as the entry of v in $\mathbf{d}$, and for a subset $U \subseteq V$, define $\mathbf{d}(U) := \sum_{v \in U} \mathbf{d}(v)$.

We now introduce the concept of an expander "weighted" by *demands* on the vertices.

Definition III.7 ($(\phi, \mathbf{d})$-expander)**.** *Consider a weighted, undirected graph $G = (V, E)$ with edge weights w and a vector $\mathbf{d} \in \mathbb{R}^V_{\geq 0}$ of non-negative "demands" on the vertices. The graph G is a $(\phi, \mathbf{d})$-expander if for all subsets $S \subseteq V$,*

$$\frac{w(\partial S)}{\min\{\mathbf{d}(S), \mathbf{d}(V \setminus S)\}} \geq \phi.$$

Intuitively, to capture the intersection of a set with U, we will place demand λ at each vertex $v \in U$, where λ is the weight of the min-cut, and demand 0 at the remaining vertices. We now state the deterministic algorithm of [17] that computes our desired expander decomposition.

Theorem III.8 ($(\phi, \mathbf{d})$-expander decomposition algorithm)**.** *Fix any constant $\epsilon > 0$ and any parameter $\phi > 0$. Given a weighted, undirected graph $G = (V, E)$ with edge weights w and a non-negative demand vector $\mathbf{d} \in \mathbb{R}^V_{\geq 0}$ on the vertices, there is a deterministic algorithm running in $O(m^{1+\epsilon})$ time that partitions V into subsets $V_1, \ldots, V_\ell$ such that*

1) *For each $i \in [\ell]$, define the demands $\mathbf{d}_i \in \mathbb{R}^{V_i}_{\geq 0}$ as $\mathbf{d}_i(v) = \mathbf{d}(v) + w(E(\{v\}, V \setminus V_i))$ for all $v \in V_i$. Then, the graph $G[V_i]$ is a $(\phi, \mathbf{d}_i)$-expander.*
2) *The total weight $w(E(V_1, \ldots, V_\ell))$ of inter-cluster edges is $B\phi\mathbf{d}(V)$ where $B = (\lg n)^{O(1/\epsilon^4)}$.*

Sparsification Algorithm. Let $\tilde{\lambda} \in [\lambda, 3\lambda]$ be a 3-approximation to the min-cut λ, which can be computed in deterministic $\tilde{O}(m)$ time using the $(2 + \delta)$-approximation algorithm of Matula (for any $\delta > 0$) [18]. Set $\phi := 1/(C \log^C n)$ for a sufficiently large constant $C > 0$, and let $\epsilon > 0$ be the constant fixed by Theorem I.1. We apply Theorem III.8 to G with parameters ϵ, ϕ and the demand vector $\mathbf{d} \in \mathbb{R}^V_{\geq 0}$ satisfying $\mathbf{d}(v) = \tilde{\lambda}$ for all $v \in U$ and $\mathbf{d}(v) = 0$ for all $v \in V \setminus U$. Observe that $\mathbf{d}(V) = |U| \cdot \tilde{\lambda} \leq |U| \cdot 3\lambda$. Let $V_1, \ldots, V_\ell \subseteq V$ be the output, and for each $i \in [\ell]$, define $U_i := V_i \cap U$.

We now describe the procedure to select the subset $U' \subseteq U$. We say that a cluster V_i is *trivial* if $U_i = \emptyset$, *small* if $1 \leq |U_i| \leq 1/\phi^2$, and *large* if $|U_i| > 1/\phi^2$. The algorithm for selecting the set U' is simple:

- for each trivial cluster, do nothing;
- for each small cluster V_i, add an arbitrary vertex of U_i to U';
- for each large cluster V_j, add $1 + 1/\phi$ arbitrary vertices of U_j to U'.

Size Bound: First, we prove the desired size bound of the sparsified set U', which is one part of Lemma III.6.

Claim III.9. *There are at most $B\phi|U|$ many clusters; that is, $\ell \leq B\phi|U|$ where $B = (\lg n)^{O(1/\epsilon^4)}$.*

Proof: Since λ is the min-cut of graph G, each cluster V_i has $w(\partial V_i) \geq \lambda$, so the total weight of inter-cluster edges is at least $\ell\lambda/2$. By the guarantee of Theorem III.8, the total weight of inter-cluster edges is at most

$$B\phi\,\mathbf{d}(V) = B\phi|U|\tilde{\lambda} \leq B\phi|U|\lambda,$$

where $B = (\lg n)^{O(1/\epsilon^4)}$. Putting these together gives $\ell \leq B\phi|U|$ as desired. ∎

Corollary III.10. *There exists a constant $C = O(1/\epsilon^4)$ such that if $\phi \leq 1/(C\log^C n)$, then the set U' constructed by the sparsification algorithm satisfies $|U'| \leq |U|/2$.*

Proof: There are at most $B\phi|U|$ small clusters by Claim III.9. Also, there are at most $\phi^2|U|$ large clusters since each large cluster has at least ϕ^2 vertices in U. This gives

$$\begin{aligned}|U'| &\leq B\phi|U| + \phi^2|U|\cdot(1+1/\phi)\\ &\leq O(B\phi|U|) \leq \phi|U|\cdot\frac{C}{2}\log^C n\end{aligned}$$

for an appropriate constant $C = O(B) = O(1/\epsilon^4)$. Since $\phi \leq 1/(C\log^C n)$, we have

$$|U'| \leq \phi|U|\cdot\frac{C}{2}\log^C n \leq |U|/2.$$

∎

Hitting Both Sides of the Min-cut: Now, we prove the "hitting" property of the sparsified set U' in Lemma III.6, namely the guarantee that $S_i \cap U' \neq \emptyset$ for both $i = 1, 2$.

The claim below says that the min-cut (A, B) cannot cut too "deeply" into the sets U_i. In particular, if a set U_i is large (say, $|U_i| \gg 1/\phi$), then the min-cut cannot cut U_i evenly in the sense that $|U_i \cap A| \approx |U_i \cap B|$; instead, we either have $|U_i \cap A| \ll |U_i \cap B|$ or $|U_i \cap A| \gg |U_i \cap B|$.

Claim III.11. *For any cut (A, B) of G, we have*

$$\sum_{i\in[\ell]} \min\{|U_i \cap A|, |U_i \cap B|\} \leq \frac{w(E(A,B))}{\phi\lambda},$$

where $U_i := V_i \cap U$ for $i \in [\ell]$.

Proof: Since $G[V_i]$ is a $(\phi, \mathbf{d}_i)$-expander, and since $\mathbf{d}_i(S) \geq \mathbf{d}(S) = |U\cap S|\cdot\tilde{\lambda} \geq |U\cap S|\cdot\lambda$ for all subsets $S \subseteq V_i$, we have

$$\begin{aligned}&\frac{w(E(V_i\cap A, V_i\cap B))}{\min\{|U\cap(V_i\cap A)|\cdot\lambda, |U\cap(V_i\cap B)|\cdot\lambda\}}\\ &\geq \frac{w(E(V_i\cap A, V_i\cap B))}{\min\{\mathbf{d}_i(U_i\cap A), \mathbf{d}_i(U_i\cap B)\}} \geq \phi.\end{aligned}$$

This means that

$$\begin{aligned}&\min\{|U_i\cap A|\cdot\lambda, |U_i\cap B|\cdot\lambda\}\\ &= \min\{|U\cap(V_i\cap A)|\cdot\lambda, |U\cap(V_i\cap B)|\cdot\lambda\}\\ &\leq \frac{w(E(U_i\cap A, U_i\cap B))}{\phi}.\end{aligned}$$

Since $E(V_i\cap A, V_i\cap B)$ is contained in $E(A,B)$ and is disjoint over all i, we have

$$\sum_{i\in[\ell]} w(E(V_i\cap A, V_i\cap B)) \leq w(E(A,B)).$$

Putting things together,

$$\begin{aligned}&\sum_{i\in[\ell]}\min\{|U_i\cap A|, |U_i\cap B|\}\\ &\leq \frac{1}{\lambda}\sum_{i\in[\ell]}\frac{w(E(V_i\cap A, V_i\cap B))}{\phi}\\ &\leq \frac{w(E(A,B))}{\phi\lambda}.\end{aligned}$$

∎

We say that a cut C *cuts* a cluster V_i if both $C \cap V_i$ and $V_i \setminus C$ are non-empty. The next claim states that the min-cut can only cut a few clusters V_i, i.e., only a few clusters V_i overlap both sides of the min-cut. This implies that for the sets $U_i \subseteq V_i$ in particular, all but a few of them satisfy $U_i \cap A = \emptyset$ or $U_i \cap B = \emptyset$.

Claim III.12. *Let C be one side of a min-cut (i.e., $w(\partial C) = \lambda$). Then, C cuts at most $(1 + 1/\phi)$ clusters V_i.*

Proof: Suppose for contradiction that C cuts more than $(1 + 1/\phi)$ clusters. Fix a cluster V_i that is cut, and let A_i and B_i be $C \cap V_i$ and $V_i \setminus C$ (possibly swapped) so that $w(E(A_i, V\setminus V_i)) \leq w(E(B_i, V\setminus V_i))$. The edges $E(A_i, B_i)$ are contained in ∂C, and across different clusters V_i that are cut, the edges $E(A_i, B_i)$ are disjoint, so

$$\sum_i w(E(A_i, B_i)) \leq w(\partial C) = \lambda.$$

Since C cuts more than $(1 + 1/\phi)$ clusters, there exists a cluster V_i with

$$w(E(A_i, B_i)) < \frac{w(\partial C)}{1+1/\phi} = \frac{\lambda}{1+1/\phi}.$$

For all subsets $S \subseteq V_i$, we have

$$\mathbf{d}_i(S) \geq \sum_{v\in S} w(E(\{v\}, V\setminus V_i)) = w(E(S, V\setminus V_i)).$$

Since $G[V_i]$ is a $(\phi, \mathbf{d}_i)$-expander,

$$\begin{aligned} &w(E(A_i, B_i)) \\ &\quad \geq \phi \cdot \min\{\mathbf{d}_i(A_i), \mathbf{d}_i(B_i)\} \\ &\quad \geq \phi \cdot \min\{w(E(A_i, V \setminus V_i)), w(E(B_i.V \setminus V_i))\} \\ &\quad = \phi \cdot w(E(A_i, V \setminus V_i)). \end{aligned}$$

Consider the cut ∂A_i, which satisfies

$$\begin{aligned} w(\partial A_i) &= w(E(A_i, B_i)) + w(E(A_i, V \setminus V_i)) \\ &\leq w(E(A_i, B_i)) + \frac{1}{\phi} w(E(A_i, B_i)) \\ &= \left(1 + \frac{1}{\phi}\right) w(E(A_i, B_i)) < \lambda, \end{aligned}$$

contradicting the fact that C is the min-cut. ∎

Finally, we prove the "hitting" property of the sparsified set U'. This, along with Corollary III.10, finishes the proof of Lemma III.6.

Lemma III.13. *Suppose that U is $(1+1/\phi)^3$-balanced with witness (S_1, S_2). Then, for the set U' constructed by the sparsification algorithm, we have $S_i \cap U' \neq \emptyset$ for both $i = 1, 2$.*

Proof: For each cluster V_i, by Claim III.11,

$$\min\{|U_i \cap A|, |U_i \cap B|\} \leq \frac{w(E(A, B))}{\phi\lambda} \leq \frac{1}{\phi}.$$

In other words, either $|S_1 \cap U_i| \leq 1/\phi$ or $|S_2 \cap U_i| \leq 1/\phi$. Call a cluster V_i:

1) *white* if $S_1 \cap U_i = \emptyset$ (i.e., $U_i \subseteq S_2$).
2) *light gray* if $0 < |S_1 \cap U_i| \leq |S_2 \cap U_i| < |U_i|$, which implies that $0 < |S_1 \cap U_i| \leq 1/\phi$.
3) *dark gray* if $0 < |S_2 \cap U_i| < |S_1 \cap U_i| < |U_i|$, which implies that $0 < |S_2 \cap U_i| \leq 1/\phi$.
4) *black* if $S_2 \cap U_i = \emptyset$ (i.e., $U_i \subseteq S_1$).

Every cluster must be one of the four colors, and by Claim III.12, there are at most $(1+1/\phi)$ (light or dark) gray clusters since $U_i \cap S_1, U_i \cap S_2 \neq \emptyset$ implies that S_1 cuts cluster V_i. Note that since we are only considering clusters V_i such that $U_i \neq \emptyset$, it must be that for a white cluster, we have $|S_2 \cap U_i| \neq \emptyset$, and similarly, for a black cluster, we have $|S_1 \cap U_i| \neq \emptyset$. There are now a few cases:

1) There are no large clusters. In this case, if there is at least one white and one black small cluster, then the vertices from these clusters added to U' are in S_2 and S_1, respectively. Otherwise, assume w.l.o.g. that there are no black clusters. Since there are at most $(1+1/\phi)$ gray clusters in total, $|S_1 \cap U| \leq (1 + 1/\phi) \cdot 1/\phi^2$, contradicting our assumption that $\min\{|S_1 \cap U|, |S_2 \cap U|\} \geq (1 + 1/\phi)^3$.
2) There are large clusters, but all of them are white or light gray. Let V_i be a large white or light gray cluster. Since we select $1 + 1/\phi$ vertices of U_i, and $|S_1 \cap U_i| = \min\{|S_1 \cap U_i|, |S_2 \cap U_i|\} \leq 1/\phi$, we must select at least one vertex not in S_1. Therefore, $S_2 \cap U' \neq \emptyset$. If there is at least one black cluster, then the selected vertex in there is in U', so $S_1 \cap U' \neq \emptyset$ too, and we are done.
   So, assume that there is no black cluster. Since all large clusters are light gray (or white), $|S_1 \cap U_i| \leq 1/\phi$ for all large clusters V_i. Moreover, by definition of small clusters, $|S_1 \cap U_i| \leq |U_i| \leq 1/\phi^2$ for all small clusters V_i. Since there are at most $(1 + 1/\phi)$ gray clusters by Claim III.12,
   $$\begin{aligned} |S_1 \cap U| &= \sum_{i:V_i \text{ small}} |S_1 \cap U_i| + \sum_{i:V_i \text{ large}} |S_1 \cap U_i| \\ &\leq \left(1 + \frac{1}{\phi}\right) \cdot \frac{1}{\phi^2} + \left(1 + \frac{1}{\phi}\right) \cdot \frac{1}{\phi} \\ &= 2\left(1 + \frac{1}{\phi}\right) \cdot \frac{1}{\phi} < \left(1 + \frac{1}{\phi}\right)^3, \end{aligned}$$
   a contradiction.
3) There are large clusters, but all of them are black or dark gray. This is symmetric to case (2) above with S_1 replaced with S_2.
4) There is at least one black or dark gray large cluster V_i, and at least one white or light gray large cluster V_j. In this case, since we select $1+1/\phi$ vertices of U_i and $|S_2 \cap U_i| = \min\{|S_1 \cap U_i|, |S_2 \cap U_i||\} \leq 1/\phi$, we must select at least one vertex in S_1. Similarly, we must select at least one vertex in U_j that is in S_2.

∎

IV. Conclusion

In this paper, we gave a deterministic algorithm for the min-cut problem in undirected graphs that uses poly-logarithmic max-flow computations plus $m^{1+\epsilon}$ time for any constant ϵ. Our main new tool is the isolation cut lemma which we hope will be useful for other problems in graph connectivity as well. The main open question left by our work is to obtain a deterministic $\tilde{O}(m)$-time algorithm for the min-cut problem on undirected graphs. Such a result is currently known only for simple graphs [19], [12].

Acknowledgments

JL was supported in part by NSF award CCF-1907820. DP was supported in part by NSF grant CCF-1955703 and an NSF CAREER Award CCF-1750140.

References

[1] R. E. Gomory and T. C. Hu, "Multi-terminal network flows," *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 4, pp. 551–570, 1961.

[2] J. Hao and J. B. Orlin, "A faster algorithm for finding the minimum cut in a graph," in *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1992, pp. 165–174.

[3] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *J. ACM*, vol. 35, no. 4, pp. 921–940, 1988. [Online]. Available: https://doi.org/10.1145/48014.61051

[4] H. Nagamochi and T. Ibaraki, "Computing edge-connectivity in multigraphs and capacitated graphs," *SIAM Journal on Discrete Mathematics*, vol. 5, no. 1, pp. 54–66, 1992.

[5] ——, "A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph," *Algorithmica*, vol. 7, no. 5&6, pp. 583–596, 1992. [Online]. Available: https://doi.org/10.1007/BF01758778

[6] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM (JACM)*, vol. 44, no. 4, pp. 585–591, 1997.

[7] D. R. Karger, "Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm." in *SODA*, vol. 93, 1993, pp. 21–30.

[8] D. R. Karger and C. Stein, "A new approach to the minimum cut problem," *Journal of the ACM (JACM)*, vol. 43, no. 4, pp. 601–640, 1996.

[9] D. R. Karger, "Minimum cuts in near-linear time," *J. ACM*, vol. 47, no. 1, pp. 46–76, 2000. [Online]. Available: http://dx.doi.org/10.1145/331605.331608

[10] H. N. Gabow, "A matroid approach to finding edge connectivity and packing arborescences," *Journal of Computer and System Sciences*, vol. 50, no. 2, pp. 259–273, 1995.

[11] K.-i. Kawarabayashi and M. Thorup, "Deterministic edge connectivity in near-linear time," *Journal of the ACM (JACM)*, vol. 66, no. 1, pp. 1–50, 2018.

[12] M. Henzinger, S. Rao, and D. Wang, "Local flow partitioning for faster edge connectivity," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, 2017, pp. 1919–1938. [Online]. Available: https://doi.org/10.1137/1.9781611974782.125

[13] Y. P. Liu and A. Sidford, "Faster divergence maximization for faster maximum flow," *arXiv preprint arXiv:2003.08929*, 2020.

[14] A. V. Goldberg and S. Rao, "Beyond the flow decomposition barrier," *Journal of the ACM (JACM)*, vol. 45, no. 5, pp. 783–797, 1998.

[15] N. Alon, R. Yuster, and U. Zwick, "Color-coding," *Journal of the ACM (JACM)*, vol. 42, no. 4, pp. 844–856, 1995.

[16] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized algorithms*. Springer, Cham, 2015. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-21275-3

[17] J. Chuzhoy, Y. Gao, J. Li, D. Nanongkai, R. Peng, and T. Saranurak, "A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond," *arXiv preprint arXiv:1910.08025*, 2019.

[18] D. W. Matula, "A linear time 2+ ε approximation algorithm for edge connectivity," in *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, 1993, pp. 500–504.

[19] K. Kawarabayashi and M. Thorup, "Deterministic edge connectivity in near-linear time," *J. ACM*, vol. 66, no. 1, pp. 4:1–4:50, 2019. [Online]. Available: https://doi.org/10.1145/3274663