

Near-Optimal Streaming Approximation for Max-DICUT in Sublinear Space Using Two Passes

Santhoshini Velusamy 

David R. Cheriton School of Computer Science, University of Waterloo, Canada

Abstract

The Max-DICUT problem has emerged as a canonical problem for understanding the approximability of constraint satisfaction problems in the streaming model. A seminal result of Kapralov and Krachun [STOC 2019] shows that it is impossible to beat $1/2$ -approximation for Max-DICUT in sublinear space in the single-pass streaming setting, even on bounded-degree graphs. In a recent work, Saxena, Singer, Sudan, and Velusamy [SODA 2025] prove that the above lower bound is tight by giving a single-pass algorithm for bounded-degree graphs that achieves $(1/2 - \epsilon)$ -approximation in sublinear space, for every constant $\epsilon > 0$. For arbitrary graphs of unbounded degree, they give an $O(1/\epsilon)$ -pass $O(\log n)$ space algorithm. Their work left open the question of obtaining $1/2$ -approximation for arbitrary graphs in the single-pass setting in sublinear space. We make progress towards this question and give a two-pass algorithm that achieves $(1/2 - \epsilon)$ -approximation in sublinear space, for every constant $\epsilon > 0$.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Design and analysis of algorithms; Theory of computation $\rightarrow$ Streaming, sublinear and near linear time algorithms; Theory of computation $\rightarrow$ Streaming models

Keywords and phrases Streaming algorithms, Approximation algorithms, Graph algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2026.4

Category APPROX

Related Version *Full Version*: <https://arxiv.org/abs/2512.19521>

Funding *Santhoshini Velusamy*: Supported in part by NSF award CCF-2348475 when the author was affiliated with Toyota Technological Institute at Chicago.

Acknowledgements Part of the work was conducted when the author was visiting the Simons Institute for the Theory of Computing as a research fellow in the Sublinear Algorithms program. The author thanks Raghuvansh Saxena, Noah Singer, and Madhu Sudan for collaborating on earlier investigations into this problem. The author thanks Aditya Bhaskara for discussions related to [5]. The author also thanks Sergio Thoumi for pointing out typos in a previous version.

1 Introduction

The maximum directed cut problem (Max-DICUT) is the analogue of the maximum cut (Max-CUT) problem for directed graphs. In particular, given a directed graph $G(V, E)$, the Max-DICUT problem asks to find an ordered bipartition (a “left” set L and a “right” set R) of the vertex set V that maximizes the number of edges that go from L to R . In contrast, in Max-CUT, we measure all the edges that cross L and R . Both Max-CUT and Max-DICUT are NP Hard and so, we study algorithms that approximately solve these problems. An α -approximation algorithm is one that outputs a solution that has a value which is at least α fraction of the optimum. Besides being fundamental problems in their own right, Max-CUT and Max-DICUT also belong to a large class of combinatorial optimization problems called the constraint satisfaction problems (CSPs) which capture several fundamental optimization problems in computer science. CSPs are well studied in theoretical computer science, owing in



© Santhoshini Velusamy;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2026).

Editors: Mohit Singh and Tom Gur; Article No. 4; pp. 4:1–4:23



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

part to their structure which allows for finite characterizations of the infinite set of problems they contain, in several computational settings. Their study has led to the discovery of several important algorithmic and complexity-theoretic tools including SDP approximations, the PCP theorem, and Unique Games.

In many computational settings, Max-CUT has successfully served as a canonical problem in the study of CSPs. For example, in the polynomial time setting, Goemans and Williamson [14] gave the first SDP-based approximation algorithm for Max-CUT. And on the lower bound side, conditioned on the Unique Games conjecture, Khot, Kindler, Mossell, and O'Donnell [20] constructed tight integrality gap instances showing that the Goemans-Williamson SDP for Max-CUT is optimal. In a breakthrough follow-up work, Raghavendra [22] generalized both these works to all CSPs and gives an optimal SDP-based characterization for the approximability of every CSP in polynomial time. However, in the streaming setting, Max-CUT does not capture all CSPs.

In the single-pass streaming model of computation, the constraints of a CSP instance (or edges of a graph in the case of Max-CUT, Max-DICUT) arrive in a stream. The streaming algorithm is allowed one pass over this stream and is required to compute an approximation to the optimum *value* of the instance, using limited space. The amount of space available to the algorithm is typically parameterized by n , the number of variables in the instance (or vertices of the input graph in the case of Max-CUT, Max-DICUT). For every CSP, there exists a trivial sampling-based algorithm [1, 28] that can achieve $(1 - \epsilon)$ -approximation using $\tilde{O}(n)$ space. Therefore, the interesting regime is when the algorithm is allowed only $o(n)$ space. Surprisingly, Max-CUT has no non-trivial approximation in this regime! In particular, $1/2$ -approximation is trivial (just output $1/2$ for every input graph) since any graph has a bipartition with at least half the edges crossing the bipartition. A series of works [17, 18, 19], culminating with the breakthrough result of Kapralov and Krachun [19] show that for any constant $\epsilon > 0$, any single-pass streaming algorithm that achieves $(1/2 + \epsilon)$ -approximation for Max-CUT requires $\Omega(n)$ space. A follow-up work due to Chou, Golovnev, Sudan, Velingker, and Velusamy [7] proves optimal inapproximability results for multiple CSPs in $o(n)$ space, generalizing techniques from the Max-CUT lower bound [19].

On the algorithmic side, the first non-trivial approximation algorithm was discovered by Guruswami, Velingker, and Velusamy [16] for Max-DICUT. While Max-DICUT has a trivial $1/4$ -approximation, they gave a single-pass algorithm that achieves $(2/5 - \epsilon)$ -approximation using only $O(\log n)$ space, for every constant $\epsilon > 0$. Chou, Golovnev, and Velusamy [10] later achieved $(4/9 - \epsilon)$ -approximation in $O(\log n)$ space. They also prove that the $4/9$ factor is optimal in $o(\sqrt{n})$ space, generalizing the techniques from the $\Omega(\sqrt{n})$ space lower bound of Max-CUT due to Kapralov, Khanna, and Sudan [17]. They further generalized their results to prove a $O(\log n)$ vs $\Omega(\sqrt{n})$ space dichotomy theorem for every Max-2CSP. This served as the basis for their subsequent works with Sudan [8, 9] which prove a $O(\log^3 n)$ vs $\Omega(\sqrt{n})$ space dichotomy characterization for every CSP.¹ Saxena, Singer, Sudan, and Velusamy [24] show that the $\Omega(\sqrt{n})$ lower bound in [10] is tight and give a 0.485 -approximation algorithm for Max-DICUT in $\tilde{O}(\sqrt{n})$ space. In a later work [25], the same authors also show that it is possible to achieve $(1/2 - \epsilon)$ -approximation for *bounded-degree* (i.e., maximum degree bounded by a constant) graphs in $O(n^{1-\Omega(1)})$ space, for every $\epsilon > 0$. A trivial reduction from Max-CUT shows that the factor $1/2$ is also optimal for Max-DICUT in $o(n)$ space.

¹ Their dichotomy characterization is for *dynamic* streaming algorithms which allow deletions and can process arbitrarily long streams.

In the multi-pass setting, a streaming algorithm is allowed to do multiple passes over the same stream of constraints. Assadi and Nagargoje [3] show that any p -pass algorithm for Max-CUT achieving $(1 - \epsilon)$ -approximation requires at least $\Omega(n^{1-O(\epsilon p)})$ space. A recent breakthrough result of Fei, Minzer, and Wang [11] shows that for any $\epsilon > 0$, any p -pass algorithm for Max-CUT (also Max-DICUT) achieving $(1/2 + \epsilon)$ -approximation requires at least $\Omega(n^{1/3}/p)$ space. In contrast, for *dense* instances of average degree at least n^δ , a two-pass streaming algorithm due to Bhaskara, Daruki, and Venkatasubramanian [5], achieves $(1 - \epsilon)$ -approximation for Max-CUT (also Max-DICUT) using $\tilde{O}(n^{1-\delta})$ space. An algorithm due to Saxena, Singer, Sudan, and Velusamy achieves $(1/2 - \epsilon)$ -approximation on arbitrary graphs in $O(1/\epsilon)$ passes using $O(\log n)$ space [25]. In a prior work [23], the same authors give a two-pass algorithm that achieves 0.485-approximation using $O(\log n)$ space. A recent result due to Fei, Minzer, and Wang [12] gives a dichotomy characterization in the multi-pass setting based on the basic LP relaxation for every CSP. In particular, they show that the techniques from [27, 25] can be combined with the distributed algorithms of Yoshida [29] to obtain a streaming algorithm that solves the LP relaxation for every CSP in $O(1/\epsilon^2)$ passes and $O(\log n)$ space. And to beat the approximation achieved by the basic LP, any p -pass streaming algorithm requires at least $\Omega(n^{1/3}/p)$ space.

The connection between streaming algorithms and LP relaxations was also identified in a concurrent work by Singer, Tulsiani, and Velusamy [26]. They conjecture that for every CSP, there must be a single-pass $o(n)$ space streaming algorithm that solves its LP relaxation, and any algorithm beating this factor must require at least $\Omega(n)$ space. For Max-DICUT, its basic LP achieves 1/2-approximation and so their conjecture states that there is a single-pass algorithm that achieves 1/2-approximation in $o(n)$ space. In this work, we make progress towards this conjecture and prove it in the two-pass setting.

► **Theorem 1.1 (Main result).** *For every $\epsilon > 0$, there is a two-pass streaming algorithm that achieves $(1/2 - \epsilon)$ -approximation for Max-DICUT on any n -vertex graph using space $O(n^{1-\Omega_\epsilon(1)})$.*

2 Technical overview

In this section, we present a technical overview of the proof of our main result.

There are three main ingredients to our proof: 1) a single-pass $(1/2 - \epsilon)$ -approximation algorithm of Saxena, Singer, Sudan, and Velusamy [25] for bounded-degree graphs that uses $O(n^{1-\Omega(1)})$ space, 2) Trevisan's reduction [27] from unbounded-degree to bounded-degree graphs that approximately preserves the Max-DICUT value, and 3) Bhaskara, Daruki, and Venkatasubramanian's [5] two-pass algorithm that achieves $(1 - \epsilon)$ -approximation on graphs of average degree at least n^δ using $\tilde{O}(n^{1-\delta})$ space. The following theorems formally state their results.

► **Theorem 2.1 ([25]).** *For every $d \in \mathbb{N}$ and $\epsilon > 0$, there is a single-pass streaming algorithm that achieves $(1/2 - \epsilon)$ -approximation for Max-DICUT on any n -vertex graph of maximum degree at most d using space $O(n^{1-\Omega(1)})$.*

► **Theorem 2.2 ([27]).** *For any directed graph $G(V, E)$ on n vertices and m edges, and $0 < \epsilon \leq 1$, there exists a (random) directed graph $\bar{G}(\bar{V}, \bar{E})$ on $2m$ vertices and $O(m/\epsilon^2)$ edges such that with probability at least $5/6$, $\text{Max-DICUT}(G) - \epsilon' \leq \text{Max-DICUT}(\bar{G}) \leq \text{Max-DICUT}(G) + \epsilon$, and the maximum degree of a vertex in $\bar{G}$ is at most $O(1/\epsilon^2)$.*

► **Theorem 2.3** ([5]). *For every $\epsilon, \delta \in (0, 1]$, there is a two-pass streaming algorithm that achieves $(1 - \epsilon)$ -approximation for Max-DICUT on any n -vertex graph of average degree at least n^δ , using space $\tilde{O}(n^{1-\delta})$.*

At a high level, our idea is to reduce the input graph which could potentially have an unbounded maximum degree to a bounded-degree graph using Trevisan’s reduction [27], and then execute the [25] algorithm for bounded-degree graphs. However, there are challenges to this naive approach. Firstly, Trevisan’s reduction is not a small-space reduction and requires the knowledge of the degree of every vertex in the graph. This would be too expensive to store in the streaming setting. Secondly, the reduction produces a graph with $2m$ vertices, which could be $\Theta(n^2)$ in the worst case, hence the [25] algorithm could require $\Omega(n)$ space. We deal with the second challenge by running the [25] algorithm only when the input graph is sufficiently sparse, i.e., has at most $n^{1+\delta}$ edges, for some suitably chosen δ so that the space usage is still at most $O(n^{1-\Omega(1)})$. And when the input instance is dense, we run the [5] algorithm. Thus, the space usage is always at most $O(n^{1-\Omega(1)})$. To deal with the first challenge, we implement the [25] algorithm in a non black-box manner. We do not produce the entire bounded-degree graph from the Trevisan reduction and instead only sample parts of this graph that are required for the [25] algorithm. Below, we give a high-level overview of both these algorithms and our implementation of them.

Trevisan reduction. The reduction is quite simple. Say $G(V, E)$ is an unweighted directed graph on n vertices and m edges, with possible multi-edges but no self-loops. Given G , Trevisan first creates a weighted graph $G'(V', E')$: for every vertex $v \in V$, he adds $\deg(v)$ many “copies” of it, labeled $(v, 0), \dots, (v, \deg(v) - 1)$, to V' . For every edge $u \rightarrow v \in E$, and $i \in [\deg(u)], j \in [\deg(v)]$,² he adds an edge $(u, i) \rightarrow (v, j)$ of weight $\frac{1}{\deg(u)\deg(v)}$ to E' . It is not hard to see that the Max-DICUT values of G and G' are equal.³ He then samples a random graph $\tilde{G}$ based on G' in the following way. The vertex set of $\tilde{G}$ is V' and he samples the edge set as follows: he sets a parameter $d = O(1/\epsilon^2)$ and samples dm edges independently (with repetition) from G' , where in each sample, an edge in E' is sampled with probability proportional to its weight. He finally deletes a minimal set of edges from $\tilde{G}$ so that the degree of every vertex is at most $O(d)$. He argues that with high probability, at most an ϵ fraction of the edges are deleted, and that the Max-DICUT value is preserved up to an additive ϵ factor.

SSSV algorithm. Saxena, Singer, Sudan, and Velusamy make a key observation that on bounded-degree graphs, in order to simulate a distributed algorithm for Max-DICUT, it suffices to estimate certain neighborhood distributions of the input graph (see Section 3.1 for more details). They show that such distributions can be estimated from the induced subgraph of a randomly sampled subset of the vertex set. Finally, based on a previous distributed algorithm of Censor-Hillel, Levy, and Shachnai [6], they give a distributed algorithm that achieves $(1/2 - \epsilon)$ -approximation for Max-DICUT, and efficiently simulate it in sublinear space in the single-pass setting using the above techniques.

² For a natural number $n \in \mathbb{N}$, we define $[n] = \{0, \dots, n - 1\}$. Note that this is slightly different from the standard definition.

³ For every $u \in V$ and $i, i' \in [\deg(u)]$, the copies (u, i) and (u, i') have identical neighborhood structure. Hence, for any fixed assignment of the remaining vertices, the DICUT value is locally maximized by placing all copies of u on the same side of the cut.

Putting them together (a three-pass algorithm). Since the [25] algorithm only requires the induced subgraph of a randomly sampled subset of the vertex set, we do not need to do the Trevisan reduction on the entire input graph. Instead, we directly sample a random subset of vertices from the bounded-degree graph, i.e., from $V' = \{(v, i) : v \in V; i \in [\deg(v)]\}$, and perform a *partial* Trevisan reduction on their *parent* vertices. To sample from V' , we first map every edge $e \in E$, based on its position in the stream, to two unique vertices in V' , i.e., no two edges map to the same vertex. In particular, an edge $u \rightarrow v$ is mapped to (u, i) and (v, j) iff it is the i -th edge (resp. j -th edge) incident on u (resp. v) in the stream. Thus, as the edges arrive in the stream, we can sample from V' .⁴ Once the vertices are sampled, we have to sample the induced subgraph on these vertices. The original Trevisan reduction performs a *global* sampling of the edges, where every edge is sampled with probability proportional to its weight. In order to perform such a sampling, one would have to compute the weights on all the edges, which is infeasible in the streaming setting. We therefore modify the reduction to a *local* sampling procedure, where for every edge $u \rightarrow v \in E$, we create $d = O(1/\epsilon^2)$ “copies” of it in E' and for every copy of the edge, we independently sample a uniform random copy of u and a uniform random copy of v in V' as its endpoints.⁵ We also delete *all* the edges incident on high-degree vertices and not just the minimal set of edges, as it is not clear how to compute the minimal set without generating the entire graph. We argue in Section 4 that this procedure still preserves the **Max-DICUT** value up to an additive ϵ factor with high probability. Now that we have this local procedure, we generate the induced subgraph on the sampled vertices as follows. After we sample the vertices in the first pass, we store the degrees of their parent vertices in the second pass. In the third pass, we again scan through the edges in the stream and if $u \rightarrow v$ is such that both u and v have at least one of their copies sampled, then we sample d independent copies of the edge as outlined above (we can do this since we know the degrees of u and v), and we store those edges that are in the induced subgraph. If neither u nor v have any copies in the sampled set of vertices, then we do nothing. If only one of them, say u , has sampled copies, then we sample only the endpoints corresponding to u ’s copies for the d copies of the edge. We do this to keep track of the degrees of all the sampled vertices, and in the end, delete all the edges incident on the high-degree vertices. The formal description of the procedure and the proof of correctness are described in Section 5.

Modifications to obtain a two-pass algorithm. In Section 6, we modify the above algorithm and get a two-pass algorithm by getting rid of the second pass which computes and stores the degrees of the parent vertices *exactly*. In particular, we show that our modified Trevisan reduction in Section 4 works even when we have only *approximate* estimates for the degrees of high-degree vertices. In the first pass, along with sampling vertices like in the above algorithm, we estimate the degrees of high-degree vertices using standard sampling methods. In particular, for every vertex, we sample each edge incident to it independently with probability $p = n^{-O_\epsilon(1)}$ and rescale the total number of sampled edges by $1/p$ to estimate its degree. This procedure would estimate the degrees accurately (up to a multiplicative error) for vertices of sufficiently-large polynomial degree, say n^c . We can ensure that the

⁴ There is a subtlety here. We do not have enough space to remember the ordering of the edges incident on every vertex in the input graph, which is crucial in the above mapping. Instead, we label the sampled copies of a vertex based on the order in which they are sampled by our algorithm, and argue that this relabeling does not affect the correctness of the algorithm (see Section 5 for more details).

⁵ A slightly different reduction, also amenable to streaming, was recently proposed by Fei, Minzer, and Wang in [12].

space is still sublinear because we run the Trevisan procedure only on graphs with bounded average degree. We then use these estimates to run edge sampling as in the third pass of the above algorithm. However, the copies of the edges incident on low-degree vertices will not be sampled correctly in this step. To fix this, for every vertex that is sampled in the first pass, we store the first n^c edges incident on its *parent* vertex in the second pass.⁶ Thus, we can identify parent vertices of degree less than n^c , and also compute their degrees *exactly*. In post-processing, we delete the edges incident to the copies of such vertices, and resample them according to their correct distributions (since we now have the exact degrees of the low-degree parent vertices).

BDV algorithm for Max-CUT. Finally, we give a high-level description of the two-pass algorithm for dense instances of Max-CUT in [5]. While their paper explicitly states the result only for Max-CUT, their algorithm immediately extends to Max-DICUT as well. Their main idea is to use certain non-uniform sampling methods to construct a “core-set” of *sublinear* size for the input graph, which preserves the Max-CUT value up to an additive error ϵ . A core-set of linear size can be obtained for any graph by uniformly sampling $\tilde{O}(n/\epsilon^2)$ edges [1, 28]. And for very dense graphs of average degree at least $\Omega(n)$, constant-size core-sets can be constructed [15, 2, 21]. In the polynomial density regime (i.e., average degree n^δ for $\delta < 1$), previous works [13, 4] show that if both the maximum and the minimum degrees are $\Theta(n^\delta)$, then a core-set of size $\tilde{O}(n^{1-\delta})$ exists. Bhaskara, Daruki, and Venkatasubramanian [5] extend this result to general graphs in this regime, and give a two-pass streaming algorithm to construct a core-set of size $\tilde{O}(n^{1-\delta})$.

3 Preliminaries

In this paper, any reference to a directed graph G means unweighted graphs with multi-edges but no self-loops. For a directed graph $G(V, E)$, a dicut is an ordered bipartition of the vertex set V . An edge $u \rightarrow v$ is said to belong to a dicut $L \sqcup R$ if $u \in L$ and $v \in R$. The value of the dicut, denoted by $\text{DICUT}(G, L \sqcup R)$ is the fraction of edges in G that belong to the dicut $L \sqcup R$. We define $\text{Max-DICUT}(G)$ to be $\max_{L \sqcup R} \text{DICUT}(G, L \sqcup R)$. For any function $\rho : V \rightarrow [0, 1]$, we define the expected dicut value of G under the rounding ρ , denoted by $\text{DICUT}(G, \rho)$, to be $\mathbb{E}[\text{DICUT}(G, L \sqcup R)]$, where every vertex in V belongs to L independently with probability $\rho(v)$.

For any function $f : A \rightarrow B$ that maps a domain A to a co-domain B , and $C \subseteq A$, we denote the restriction of the function to the domain C by $f|_C : C \rightarrow B$.

Distribution Notation

Let $\Delta(S)$ denote the set of all probability distributions over a finite set S . A distribution $\mu \in \Delta(S)$ assigns to each element $x \in S$ a probability $\mu(x) \geq 0$ such that $\sum_{x \in S} \mu(x) = 1$.

Total Variation Distance

The *total variation distance* between two distributions $\mu, \nu \in \Delta(S)$ is defined as

$$d_{\text{TV}}(\mu, \nu) = \frac{1}{2} \sum_{x \in S} |\mu(x) - \nu(x)|.$$

⁶ We need to be a bit careful when defining the degree thresholds and the sampling probabilities, but they can be set in a way to keep the space sublinear.

Equivalently,

$$d_{\text{TV}}(\mu, \nu) = \max_{A \subseteq S} |\mu(A) - \nu(A)|,$$

where $\mu(A) = \sum_{x \in A} \mu(x)$. The total variation distance satisfies $d_{\text{TV}}(\mu, \nu) \in [0, 1]$, with $d_{\text{TV}}(\mu, \nu) = 0$ if and only if $\mu = \nu$, and $d_{\text{TV}}(\mu, \nu) = 1$ if and only if the supports of μ and ν are disjoint.

We now list some standard concentration bounds that will be used in the rest of the paper.

► **Theorem 3.1** (Chebyshev's inequality). *Let X be a random variable with variance σ^2 . For any $a > 0$, we have*

$$\Pr[|X - \mathbb{E}[X]| > a] \leq \frac{\sigma^2}{a^2}.$$

► **Theorem 3.2** (Chernoff bound). *Let $X_1, \dots, X_n$ be independent random variables taking values in $\{0, 1\}$. Consider the sum of these random variables, $S_n = X_1 + \dots + X_n$. Let $\mu = \mathbb{E}[S_n]$. Then for $R \geq 2e\mu$,*

$$\Pr[S_n \geq R] \leq 2^{-R}.$$

► **Theorem 3.3** (Hoeffding's inequality). *Let $X_1, \dots, X_n$ be independent bounded random variables such that $a_i \leq X_i \leq b_i$. Consider the sum of these random variables, $S_n = X_1 + \dots + X_n$. Then Hoeffding's theorem states that for all $t > 0$,*

$$\Pr[|S_n - \mathbb{E}[S_n]| \geq t] \leq 2 \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

3.1 Neighborhood distributions and the simulation of local algorithms

A key contribution of Saxena, Singer, Sudan, and Velusamy in [25] is the identification of a connection between streaming and local algorithms, where they show that it is possible to estimate local neighborhood distributions of graphs in the streaming setting and use these distributions to simulate constant-round distributed algorithms. We recall some of the notations introduced in their paper and reprove their main theorem that shows that sampling vertices uniformly at random and estimating local neighborhood distributions from the induced subgraph suffices to estimate the value of a local algorithm that is executed on the original graph.

First, we formally define the LOCAL model for distributed algorithms for the Max-DICUT problem and state the result of [6, 25].

► **Definition 3.4** (The LOCAL model for distributed algorithms for Max-DICUT). *Let $G(V, E)$ be a directed graph on n vertices and m edges. G describes the topology of a communication network, where each node can communicate with both its incoming and outgoing neighbors. Communication happens in synchronous rounds, where in every round, every node does some internal computations and sends an arbitrarily long message to each of its neighbors. At the end, each node v computes a fractional assignment $\rho(v) \in [0, 1]$ and the approximation ratio achieved by the algorithm is defined to be $\text{DICUT}(G, \rho) / \text{Max-DICUT}(G)$. In randomized algorithms, the nodes also have access to an arbitrarily long private random bit string that is sampled prior to any communication between the nodes.*

► **Theorem 3.5** ([6, 25]). *For every $\epsilon > 0$, there exists a distributed algorithm for Max-DICUT in the LOCAL model that achieves $(1/2 - \epsilon)$ -approximation with probability at least $9/10$, and requires at most $1/\epsilon$ rounds of communication. It is a randomized algorithm where every node uses only an $O(\log(1/\epsilon))$ -bit private random string, and the random strings can be sampled from a pairwise independent hash family.*

Saxena, Singer, Sudan, and Velusamy [25] observed that for each vertex v , $\rho(v)$ depends only on the structure of a ball of radius $1/\epsilon$ centered at that node, and the private random strings of all the nodes in this ball. They called this the “neighborhood type” of a vertex. They observed that in order to compute $\text{DICUT}(G, \rho) = (\sum_{u \rightarrow v \in E} \rho(u)(1 - \rho(v)))/m$ with a small additive error and a constant success probability, it suffices to sample a random constant-sized subset $S \subset E$ and output $\sum_{u \rightarrow v \in S} \rho(u)(1 - \rho(v))/|S|$. This motivated them to define the notion of a neighborhood type of an edge that captures the neighborhood types of both its endpoints. Thus, estimating the distribution of the edge-neighborhood types would suffice to estimate $(\sum_{u \rightarrow v \in E} \rho(u)(1 - \rho(v)))/m$, which in turn gives a $(1/2 - \epsilon)$ -approximation to $\text{Max-DICUT}(G)$. For graphs whose maximum degree is bounded by a constant, the set of all possible edge-neighborhood types is constant sized, and hence, they could estimate their distribution efficiently in the streaming setting. In what follows, we will set up some notations to define the neighborhood types and reprove a key theorem from [25].

Let $G(V, E)$ be a directed graph and $u, v \in V$. We say that v is of distance at most ℓ from u if there exists a *path* from u to v which is a sequence of vertices $u = w_0, w_1, \dots, w_{\ell-1}, w_\ell = v \in V$ such that for each $i \in [\ell]$, $w_{i-1} \rightarrow w_i \in E$ or $w_i \rightarrow w_{i-1} \in E$. Let $\mathbf{Typ}_c^{\ell, D}$ denote the set of all isomorphism classes of doubly rooted labeled graphs where every vertex is of degree at most D , has a c -bit label, and is of distance at most ℓ from (at least) one of the roots. Consider a directed graph $G(V, E)$ whose maximum degree is at most D . Let $\mathcal{H}_n$ be the pairwise independent hash family that is used to sample the private random strings for every node in the distributed algorithm of [6, 25]. Let $h : V \rightarrow \{0, 1\}^c$ be any hash function in $\mathcal{H}_n$. For an edge $e \in E$, consider the following doubly rooted labeled graph $\mathcal{G}_e$: $\mathcal{G}_e$ is rooted at the endpoints of e and consists of all the vertices in G which are of distance at most ℓ from (at least) one of the roots, and their corresponding induced subgraph. In addition, each vertex v in $\mathcal{G}_e$ is labeled $h(v)$. The radius- ℓ neighborhood type of e with respect to G and the hash function h , $\text{nbhdtype}_{G,h}^\ell(e)$, is defined to be the representative of the isomorphism class in $\mathbf{Typ}_c^{\ell, D}$ that contains $\mathcal{G}_e$. The neighborhood edge-type distribution of G with respect to the hash function h is defined as follows.

► **Definition 3.6** (Edge-type distribution). *Let $h : V \rightarrow \{0, 1\}^c$, $\ell, D \in \mathbb{N}$ and $G = (V, E)$ be a directed graph of maximum degree D . The radius- ℓ neighborhood type distribution of G with respect to h , denoted $\text{EdgeNbhdTypeDist}_{G,h}^\ell$, is the distribution over $\mathbf{Typ}_c^{\ell, D}$ given by sampling a random $e \sim \text{Unif}(E)$ and outputting $\text{nbhdtype}_{G,h}^\ell(e)$.*

► **Corollary 3.7** ([6, 25]). *For every $\epsilon'' > 0$, there exist $\ell, c, n_0 \in \mathbb{N}$ and a function $\text{Local} : \mathbf{Typ}_c^{\ell, D} \rightarrow [0, 1]$ such that for every $n \geq n_0$, there exists a hash family $\mathcal{H}_n$ such that the following holds. Let $G = (V, E)$ be a directed graph on n vertices and of maximum degree D . Let $h \sim \text{Unif}(\mathcal{H}_n)$. Then with probability at least $9/10$,*

$$\frac{\text{Max-DICUT}(G)}{2} - \epsilon'' \leq \mathbb{E}_{T \sim \text{EdgeNbhdTypeDist}_{G,h}^\ell} [\text{Local}(T)] \leq \text{Max-DICUT}(G).$$

The following corollary implies that it suffices to compute a distribution that is close in total variation distance to $\text{EdgeNbhdTypeDist}_{G,h}^\ell$.

► **Corollary 3.8.** *If $\mathcal{D} \in \Delta(\mathbf{Typ}_c^{\ell, D})$ is such that $\text{tvd}(\mathcal{D}, \text{EdgeNbhdTypeDist}_{G,h}^\ell) \leq \epsilon''$, then*

$$\frac{\text{Max-DICUT}(G)}{2} - 2\epsilon'' \leq \mathbb{E}_{T \sim \mathcal{D}} [\text{Local}(T)] \leq \text{Max-DICUT}(G) + \epsilon''.$$

The following theorem (combined with Corollary 3.8) shows that for bounded-degree graphs, in order to achieve a $(1/2 - \epsilon)$ -approximation to Max-DICUT, it suffices to sample a sublinear number of random vertices and compute a “rescaled” edge-type neighborhood distribution of the induced subgraph.

► **Theorem 3.9** (Simplified non-streaming version of Algorithm 2 in [25]). *For every $\epsilon'' > 0, \alpha \in [0, 1), D, c, \ell \in \mathbb{N}$, there exist $\beta_0 \in (0, 1)$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and $n^{-\beta_0} \leq p \leq 1$, the following hold: Let $G(V, E)$ be a directed graph on n vertices and m edges such that the maximum degree is at most D and at most α fraction of the vertices are isolated. Let $h \in V \rightarrow \{0, 1\}^c$. Let $V' \subset V$ be a random subset where every vertex in V is sampled independently with probability p . Let $G' \subseteq G$ be the induced subgraph of G on the vertices in V' . For every $T \in \mathbf{Typ}_c^{\ell, D}$, let a_T denote the number of edges e in G' such that all the vertices in G which are at a distance of at most ℓ from at least one of the endpoints of e , are contained in V' , and $\text{nbhdtype}_{G', h|_{V'}}^\ell(e) = T$. Consider the distribution $\mathcal{D} \in \Delta(\mathbf{Typ}_c^{\ell, D})$, defined to be*

$$\mathcal{D}(T) = \frac{a_T \cdot p^{-|T|}}{\sum_{T' \in \mathbf{Typ}_c^{\ell, D}} a_{T'} \cdot p^{-|T'|}},$$

where $T \in \mathbf{Typ}_c^{\ell, D}$ and $|T|$ denotes the number of vertices in T . Then, with probability at least $9/10$, $\text{tvd}(\mathcal{D}, \text{EdgeNbhdTypeDist}_{G;h}^\ell) \leq \epsilon''$.

A formal proof of this theorem appears in the full version.

4 Reduction from unbounded-degree graphs to bounded-degree graphs

In this section, we give a generalized version of the Trevisan reduction [27] that is tailored to be implementable in the two-pass streaming setting in Section 6.

■ **Algorithm 1** Algorithm for reducing unbounded-degree graphs to bounded-degree graphs.

Parameters: $n, m \in \mathbb{N}$, $\epsilon' > 0$, $\zeta > 0$

Input:

A directed graph $G(V, E)$ on n vertices and m edges

An n -length array **approx-deg** (indexed by V) such that when $\deg(v) < n^\zeta$, we have **approx-deg**(v) = $\deg(v)$ and when $\deg(v) \geq n^\zeta$, we have $(1 - \epsilon'/100) \deg(v) \leq \text{approx-deg}[v] \leq (1 + \epsilon'/100) \deg(v)$

- 1: Initialize directed graphs $\tilde{G}(\tilde{V}, \tilde{E}), \overline{G}(\overline{V}, \overline{E})$, where $\tilde{V} = \overline{V} = \{(v, i) : v \in V, i \in [\deg(v)]\}$ and $\tilde{E} = \emptyset, \overline{E} = \emptyset$
- 2: Set a parameter $d = 80/(\epsilon')^2$
- 3: **for** every edge $u \rightarrow v \in E$ **do**
- 4: Initialize $i = 1$
- 5: **for** $i \leq d$ **do**
- 6: Sample $i_1 \sim \text{Unif}([\text{approx-deg}(u)])$
- 7: Sample $i_2 \sim \text{Unif}([\text{approx-deg}(v)])$
- 8: **if** $i_1 < \deg(u)$ and $i_2 < \deg(v)$ **then**
- 9: Add $(u, i_1) \rightarrow (v, i_2)$ to $\tilde{E}$
- 10: Increment i by 1
- 11: Copy all the edges in $\tilde{E}$ to $\overline{E}$
- 12: **for** every $v \in \overline{V}$ such that $\deg(v) > 11d$ **do**
- 13: Delete all the edges incident on v in $\overline{E}$
- 14: Output $\overline{G}(\overline{V}, \overline{E})$

► **Theorem 4.1.** *Let $n \geq 2, m \geq 10 \in \mathbb{N}$. For any directed graph $G(V, E)$ on n vertices and m edges, and $0 < \epsilon' \leq 1, \zeta > 0$, with probability at least $5/6$, Algorithm 1 outputs a directed graph $\tilde{G}(\tilde{V}, \tilde{E})$ on $2m$ vertices and $\Theta(m/(\epsilon')^2)$ edges such that $\text{Max-DICUT}(G) - \epsilon' \leq \text{Max-DICUT}(\tilde{G}) \leq \text{Max-DICUT}(G) + \epsilon'$, and the maximum degree of a vertex in $\tilde{G}$ is at most $O(1/(\epsilon')^2)$.*

Proof. Recall the construction of $\tilde{G}$ in Algorithm 1. We will first argue that the number of edges in $\tilde{E}$ is at least $md(1 - \epsilon'/8)$, with high probability. Observe that for any edge $u \rightarrow v \in E$, when we sample its i -th copy in $\tilde{G}$, the probability that either i_1 or i_2 fail to satisfy the condition in Algorithm 1 is at most $\epsilon'/50$. Therefore, applying Chernoff bound (Theorem 3.2), we conclude that with probability at least $1 - \exp(-10m)$, $|\tilde{E}| \geq md(1 - \epsilon'/8)$.

Max-DICUT value of $\tilde{G}$. Fix any ordered bipartition $L \sqcup R$ of $\tilde{V}$. For an edge $e \in E$, let $X_{e,i}$ be the indicator random variable for the event that in the i -th iteration corresponding to edge e , the condition in Algorithm 1 is satisfied and the corresponding edge in $\tilde{E}$ belongs to the dicut $L \sqcup R$. Then, the DICUT value of $\tilde{G}$ under $L \sqcup R$ is given by

$$\text{DICUT}(\tilde{G}, L \sqcup R) = \frac{\sum_{e \in E, 1 \leq i \leq d} X_{e,i}}{|\tilde{E}|}.$$

Let us define a function $\rho_{L \sqcup R} : V \rightarrow [0, 1]$ such that for $v \in V$, $\rho_{L \sqcup R}(v) = (|L \cap \{(v, i) : i \in [\deg(v)]\}|) / \deg(v)$, i.e., $\rho_{L \sqcup R}(v)$ is the fraction of copies of v in L . Then,

$$\text{DICUT}(G, \rho_{L \sqcup R}) - \epsilon'/33 \leq \frac{\sum_{e \in E, 1 \leq i \leq d} \mathbb{E}[X_{e,i}]}{dm} \leq \text{DICUT}(G, \rho_{L \sqcup R}) + \epsilon'/33.$$

This is because for every $u \rightarrow v \in E, 1 \leq i \leq d$,

$$\rho_{L \sqcup R}(u)(1 - \rho_{L \sqcup R}(v)) - \epsilon'/33 \leq X_{u \rightarrow v, i} \leq \rho_{L \sqcup R}(u)(1 - \rho_{L \sqcup R}(v)) + \epsilon'/33, \quad (4.1)$$

and

$$\text{DICUT}(G, \rho) = \frac{\sum_{u \rightarrow v} \rho(u)(1 - \rho(v))}{m}.$$

To see why Equation (4.1) is true, observe that for $i_1 \sim \text{Unif}([\text{approx-deg}[u]])$, we have

$$\rho_{L \sqcup R}(u) - \epsilon'/99 \leq \Pr[i_1 < \deg(u) \text{ and } (u, i_1) \in L] \leq \rho_{L \sqcup R}(u) + \epsilon'/99,$$

since for every $u \in V$, $\text{approx-deg}[u]$ satisfies

$$(1 - \epsilon'/100) \deg(u) \leq \text{approx-deg}[u] \leq (1 + \epsilon'/100) \deg(u),$$

Since all the edges were sampled independently and $d = 80/(\epsilon')^2$, we can apply Hoeffding's bound (Theorem 3.3) to conclude that with probability at least $1 - \exp(-4m)$,

$$\left| \frac{\sum_{e \in E, 1 \leq i \leq d} X_{e,i}}{dm} - \text{DICUT}(G, \rho_{L \sqcup R}) \right| \leq \epsilon'/4.$$

Conditioned on $|\tilde{E}| \geq md(1 - \epsilon'/8)$, we have

$$|\text{DICUT}(\tilde{G}, L \sqcup R) - \text{DICUT}(G, \rho_{L \sqcup R})| \leq \epsilon'/2.$$

Since $\text{Max-DICUT}(G) = \max_{\rho: V \rightarrow \{0,1\}} \text{DICUT}(G, \rho) = \sup_{\rho: V \rightarrow [0,1]} \text{DICUT}(G, \rho)$, taking the union bound over the 2^{2m} possible partitions of $\tilde{V}$,⁷ we conclude that with probability at least $1 - \exp(-2m)$,

$$|\text{Max-DICUT}(\tilde{G}) - \text{Max-DICUT}(G)| \leq \epsilon'/2.$$

The total number of edges incident on high-degree vertices in $\tilde{G}$. For a vertex $\tilde{v} \in \tilde{V}$, let $Y_{\tilde{v}} = \mathbb{1}[\deg(\tilde{v}) > 11d] \deg(\tilde{v})$. Then $Y = \sum_{\tilde{v} \in \tilde{V}} Y_{\tilde{v}}$ denotes the total sum of the degrees of vertices whose degree is larger than $11d$. It suffices to show that for any $\tilde{v}$, $\mathbb{E}[Y_{\tilde{v}}] < 1$. This is because applying Markov's inequality, we would get that $Y < 8m/7 < \epsilon'|\tilde{E}|/8$ with probability at least $7/8$, conditioned on $|\tilde{E}| \geq md(1 - \epsilon'/8)$. Since $\text{Max-DICUT}(\tilde{G}) \geq |\tilde{E}|/4$, the graph $\bar{G}$ obtained by deleting all the edges incident on vertices of degree larger than $11d$ in $\tilde{G}$ would satisfy $|\text{Max-DICUT}(\bar{G}) - \text{Max-DICUT}(\tilde{G})| \leq \epsilon'/2$. We also get that $|\bar{E}| \geq md(1 - \epsilon')/4$.

It remains to show that $\mathbb{E}[Y_{\tilde{v}}] < 1$. We will argue that for $d' > 11d$, $\Pr[\deg(\tilde{v}) \geq d'] \leq 2^{-d'}$ (thus, $\mathbb{E}[Y_{\tilde{v}}] \leq \sum_{d' > 11d} \frac{d'}{2^{-d'}} < 1$). Consider any vertex $\tilde{v} = (v, i) \in V$. Consider any edge $e \in E$ that is incident on v . For $1 \leq j \leq d$, let $Z_{e,j}$ be the indicator random variable for the event that in the j -th iteration corresponding to e , the condition in Algorithm 1 is satisfied and the edge that is sampled is incident on $\tilde{v}$. We have

$$\deg(\tilde{v}) = \sum_{e \in E: e \text{ is incident on } v} \sum_{1 \leq j \leq d} Z_{e,j}.$$

Observe that $\mathbb{E}[Z_{e,j}] \leq \frac{1}{\deg(v)(1-\epsilon'/100)}$. Thus, by linearity of expectation, $\mathbb{E}[\deg(\tilde{v})] \leq \frac{d \deg(v)}{\deg(v)(1-\epsilon'/100)} \leq 2d$ and since the edges were sampled independently, we can apply Chernoff bound (Theorem 3.2) to conclude that $\Pr[\deg(\tilde{v}) \geq d'] \leq 2^{-d'}$. ◀

5 Three-pass streaming algorithm for graphs of bounded average degree

In this section, for every constant $\epsilon > 0$, we give a three-pass streaming algorithm that achieves $(1/2 - \epsilon)$ -approximation on any n -vertex graph with average degree at most n^δ , using space $O(n^{1-\delta})$, for some $\delta \in (0, 1)$.

Algorithm 2 Three-pass streaming algorithm for Max-DICUT on graphs of bounded average degree.

Parameters: $n \in \mathbb{N}$, $\epsilon \in (0, 1/2)$

Input: A stream σ of edges of directed graph $G(V, E)$ on n vertices and at most $n^{1+\delta}$ edges, where $\delta = \epsilon^{25/\epsilon}$

First pass:

- 1: Maintain a counter for the number of edges m
- 2: Set a parameter $\beta \triangleq \epsilon^{20/\epsilon}$
- 3: Initialize $V', E' = \emptyset$
- 4: Initialize an n -length array **count** (indexed by V) to be 0ⁿ
- 5: **for** every edge $u \rightarrow v$ in the stream **do**

⁷ In particular, consider the Max-DICUT partition $\rho^* : V \rightarrow \{0, 1\}$ of G and the partition $L \sqcup R$ of $\tilde{V}$ where $(v, i) \in L$ iff $\rho^*(v) = 1$. By definition, $\rho_{L \sqcup R} = \rho^*$.

```

6:   Sample  $z_1, z_2 \sim \text{Bern}(n^{-\beta})$  independentlya
   a We denote by  $\text{Bern}(p)$  the Bernoulli distribution where we draw 1 with probability  $p$  and 0 with
   probability  $1 - p$ .
7:   if  $z_1 = 1$  then
8:     | Add  $(u, \text{count}[u])$  to  $V'$  and increment  $\text{count}[u]$  by 1
9:   if  $z_2 = 1$  then
10:    | Add  $(v, \text{count}[v])$  to  $V'$  and increment  $\text{count}[v]$  by 1
11:   if  $|V'| > n^{1-3\beta/4}$  then
12:     | Terminate the execution

```

Second pass:

```

13: if  $m \leq n^{1-\epsilon^{1/\epsilon}}$  then
14:   | Store all the edges and compute the Max-DICUT value.
15: else
16:   | for every  $v \in V$  such that  $\text{count}[v] > 0$  do
17:     | Compute  $\deg(v)$  in the stream
18:   |  $\forall i \in [\text{count}[v]]$ , initialize  $d(v, i) = 0$ 

```

Third pass:

```

19: Set a parameter  $d = 320/\epsilon^2$ 
20: for every edge  $u \rightarrow v$  in the stream do
21:   Initialize  $i = 1$ 
22:   for  $i \leq d$  do
23:     if  $\text{count}[u] > 0$  then
24:       | Sample  $i_1 \sim \text{Unif}([\deg(u)])$ 
25:       | if  $i_1 < \text{count}[u]$  then
26:         | Increment  $d(u, i_1)$  by 1
27:     if  $\text{count}[v] > 0$  then
28:       | Sample  $i_2 \sim \text{Unif}([\deg(v)])$ 
29:       | if  $i_2 < \text{count}[v]$  then
30:         | Increment  $d(v, i_2)$  by 1
31:     if  $\text{count}[u] > 0, \text{count}[v] > 0$  and  $i_1 < \text{count}[u], i_2 < \text{count}[v]$  then
32:       | Add  $(u, i_1) \rightarrow (v, i_2)$  to  $E'$ 
33:       | if  $|E'| > n^{1-\epsilon^{25/\epsilon}}$  then
34:         | Terminate the execution
35:     | Increment  $i$  by 1

```

Post processing steps of Algorithm 2

1. For every $(u, i) \in V'$ such that $d(u, i) > 11d = 3520/\epsilon^2$, delete every edge incident on (u, i) in E' and decrement $d(\cdot)$ each of its endpoints by 1. Finally, set $d(u, i) = 0$. Let $\mathcal{G}(V', E')$ be the directed graph on vertex set V' and edge set E' .
2. Let $\ell, c \in \mathbb{N}$ be set according to Corollary 3.7 for parameter $\epsilon'' = \epsilon/8$. Let $D = 11d$. Let $\mathcal{H}_{2m}$ be the hash family in Corollary 3.7 corresponding to directed graphs on $2m$ vertices. Sample $h \sim \text{Unif}(\mathcal{H}_{2m})$.
3. For $T \in \mathbf{Typ}_c^{\ell, D}$, let a_T denote the number of edges $e \in E'$ such that every vertex $(v, i) \in V'$ which is at a distance of at most $\ell - 1$ from at least one of the endpoints of e satisfies the property that $\deg_{\mathcal{G}}(v, i) = d(v, i)$, and $\text{nbhdtype}_{\mathcal{G}, h|_{V'}}^{\ell}(e) = T$.

4. Consider the distribution $\mathcal{D} \in \Delta(\mathbf{Typ}_c^{\ell,D})$, defined to be

$$\mathcal{D}(T) = \frac{a_T \cdot n^{\beta|T|}}{\sum_{T' \in \mathbf{Typ}_c^{\ell,D}} a_{T'} \cdot n^{\beta|T'|}},$$

where $T \in \mathbf{Typ}_c^{\ell,D}$ and $|T|$ denotes the number of vertices in T .

5. Let $\text{Local} : \mathbf{Typ}_c^{\ell,D} \rightarrow [0, 1]$ be the function in Corollary 3.7. Output $\mathbb{E}_{T \sim \mathcal{D}}[\text{Local}(T)]$.

► **Theorem 5.1.** *For every $\epsilon \in (0, 1/2)$, there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and every directed graph G on n vertices and $m \leq n^{1+\epsilon^{25/\epsilon}}$ edges, Algorithm 2 outputs a value v satisfying*

$$\frac{\text{Max-DICUT}(G)}{2} - \epsilon \leq v \leq \text{Max-DICUT}(G) + \epsilon,$$

with probability at least $2/3$, using space $n^{1-\epsilon^{O(1/\epsilon)}}$.

Proof. We first analyze the space usage of Algorithm 2 and then argue its correctness.

Space usage of Algorithm 2. It follows from Algorithm 2 that $|V'|, |E'| \leq n^{1-\epsilon^{O(1/\epsilon)}}$. In Algorithm 2, we can maintain **count** as an associative array with keys in V , thus storing only the non-zero entries in **count**, which is at most $|V'|$. We initialized **count** as an n -length array to keep the presentation simple.

Correctness of Algorithm 2. Consider the bounded-degree graph $\overline{G}$ that we constructed in Theorem 4.1 such that

$$\text{Max-DICUT}(G) - \epsilon/2 \leq \text{Max-DICUT}(\overline{G}) \leq \text{Max-DICUT}(G) + \epsilon/2.$$

We will show that with probability at least $2/3$,

$$\text{tvd}(\mathcal{D}, \text{EdgeNbhdTypeDist}_{\overline{G};h}^{\ell}) \leq \epsilon/8.$$

Therefore, combining Theorem 4.1 and Corollaries 3.7 and 3.8, the correctness of Algorithm 2 immediately follows. Let us first recall the construction of $\overline{G}$ in Algorithm 1 for the parameter $\epsilon' = \epsilon/2$, and when $\text{est-deg}[v] = \deg(v)$ for every $v \in V$. For every vertex $v \in V$, we add $\deg(v)$ many copies of it to $\overline{V}$, i.e., $\overline{V} = \{(v, i) : v \in V, i \in [\deg(v)]\}$. We fix a parameter $d = 10/(\epsilon')^2 = 40/\epsilon^2$. For every edge $u \rightarrow v \in E$, we sample a copy (u, i) of u and a copy (v, j) of v uniformly at random from $\overline{V}$, and add the edge $(u, i) \rightarrow (v, j)$ to $\overline{E}$. We repeat this process d times independently for every edge, and finally, for every vertex of degree larger than $11d$ in $\overline{V}$, we delete all the edges incident to it. We prove that with probability at least $5/6$,

$$\text{Max-DICUT}(G) - \epsilon/2 \leq \text{Max-DICUT}(\overline{G}) \leq \text{Max-DICUT}(G) + \epsilon/2.$$

For now, let us ignore the termination conditions in Algorithm 2. We will later argue that these events occur with very low probability and hence, do not affect the success probability of the algorithm by much. We claim the following.

► **Claim 5.2.** At the end of the third pass (before post processing), the following hold:

1. V' has the same distribution as sampling every vertex from $\overline{V}$ independently with probability $n^{-\beta}$, and applying a relabeling on the vertices, and

2. for $u \rightarrow v \in E$, if both u and v have copies in V' , then the copies of $u \rightarrow v$ in $\tilde{G}$ are sampled identically in Algorithms 1 and 2. If only one of u or v has a copy in V' , then the endpoints corresponding to the copies of that vertex in the copies of $u \rightarrow v$ in $\tilde{G}$ are sampled identically in Algorithms 1 and 2.

Proof. Consider the map $f : \bar{V} \rightarrow E$ defined as follows: for $(v, i) \in \bar{V}$, $f(v, i) = e$, where e is the $(i + 1)$ -th edge incident on v in the stream σ . Observe that for every edge $u \rightarrow v$, there are exactly two vertices in $\bar{V}$ that map to it, namely (u, i) and (v, j) where i, j are such that e is the $(i + 1)$ -th edge (resp. $(j + 1)$ -th edge) incident on u (resp. v) in the stream σ . Therefore, sampling every vertex in $\bar{V}$ independently with probability $n^{-\beta}$ is the same as sampling each vertex in $f^{-1}(e)$, for every edge e , independently with probability $n^{-\beta}$. This is exactly what we do in Algorithm 2. However, since we cannot maintain information about the streaming order of all the edges incident on every vertex in V , we relabel the copies of v in $\bar{V}$ based on the order in which they are sampled by our algorithm. In particular, for a vertex $v \in V$, the first copy that is sampled is labeled $(v, 0)$, the second copy that is sampled is labeled $(v, 1)$, and so on. This proves Item 1 of Claim 5.2. To see Item 2 of Claim 5.2, recall how the edges were sampled for $\tilde{G}$ in Algorithm 1. For our purposes, it suffices to consider the sampling of the copies of only those edges in E that are incident on a vertex $v \in V$ which has a copy in V' , i.e., $V' \cap \{(v, i) : i \in [\deg(v)]\} \neq \emptyset$. Algorithm 2 check for this condition and if they are satisfied, the sampling is identical to Algorithm 1 of Algorithm 1. For an edge $u \rightarrow v$, its copy is added to E' in Algorithm 2 iff the sampled copies of both its endpoints are in V' . Observe that for $(v, i) \in V'$, all the edges incident on v in E satisfy either the condition in Algorithm 2 or Algorithm 2. $\triangleleft$

Given the above claim, we consider $\tilde{G}$ where for $u \rightarrow v \in E$, if both u and v have copies in V' , then the copies of $u \rightarrow v$ are sampled according to Algorithm 2 (prior to the post-processing steps). If only one of u or v has a copy in V' , then the endpoints corresponding to the copies of that vertex in the copies of $u \rightarrow v$ are sampled according to Algorithm 2 and the other endpoints are sampled independently, according to Algorithm 1. The remaining edges are also sampled according to Algorithm 1. Finally, like in Algorithm 1, we define $\bar{G}$ as $\tilde{G}$ with edges incident on vertices of degree larger than $11d$ deleted.⁸ Observe that for $\bar{G}$ as defined above, $\mathcal{G}(V', E')$ is the induced subgraph of $\bar{G}$ on V' and furthermore, at the end of the post-processing steps of Algorithm 2, for $(v, i) \in V'$, $d(v, i)$ is degree of (v, i) in $\bar{G}$. This is true because for any vertex $(v, i) \in V'$, its degree in $\tilde{G}$ depends only on the randomness in Algorithm 2.

We are now ready to show that with probability at least $3/4$,

$$\text{Max-DICUT}(G) - \epsilon/2 \leq \text{Max-DICUT}(\bar{G}) \leq \text{Max-DICUT}(G) + \epsilon/2,$$

and

$$\text{tvd}(\mathcal{D}, \text{EdgeNbhdTypeDist}_{\bar{G};h}^\ell) \leq \epsilon/8.$$

We apply Theorems 3.9 and 4.1. In particular, for Theorem 4.1, we set the parameter $\epsilon' = \epsilon/2$. We get that with probability at least $5/6$,

- $\text{Max-DICUT}(G) - \epsilon/2 \leq \text{Max-DICUT}(\bar{G}) \leq \text{Max-DICUT}(G) + \epsilon/2$,
- the maximum degree of any vertex in $\bar{G}$ is at most $110/(\epsilon')^2 = 3520/\epsilon^2$, and
- $\bar{G}$ has at least $240m/\epsilon^2$ edges.

⁸ A key observation here is that sampling $\bar{G}$ first and then sampling V' is equivalent to sampling V' and then sampling $\bar{G}$.

Since $\overline{G}$ has exactly $2m$ vertices, it follows that at most 0.94 fraction of them are isolated. Now conditioned on this $\overline{G}$, we set the parameters of Theorem 3.9. We set $\epsilon'' = \epsilon/8$, $D = 3520/\epsilon^2$, and c, ℓ, h according to Algorithm 2, and $\alpha = 0.94$. Now set β_0 and n_0 according to the proof of Theorem 3.9. It follows from Claim 5.2 that V' is obtained by sampling every vertex in $\overline{V}$ independently with probability $n^{-\beta}$, and $\mathcal{G}(V', E')$ is the induced subgraph of $\overline{G}$ on V' . Say $2m \geq n_0$. To apply Theorem 3.9, observe that sampling $\overline{G}$ first and then sampling V' is the same as sampling V' first and then sampling $\overline{G}$, and relabeling of the copies of the vertices does not change the distribution. Thus, it remains to show the following.

1. The sampling probability $n^{-\beta}$ is at least $(2m)^{-\beta_0}$, and
2. a_T defined in the third post-processing step of Algorithm 2 is the same as the a_T defined in Theorem 3.9.

If $m \leq n^{1-\epsilon^{1/\epsilon}}$, then Algorithm 2 would simply store all the edges and compute the Max-DICUT value. So without loss of generality, we can assume that $m > n^{1-\epsilon^{1/\epsilon}}$, and in this case, $n^{-\beta} \geq (2m)^{-\beta_0}$. In Claim 5.2, we also show that for every $(v, i) \in V'$, $d(v, i)$ is the degree of (v, i) in $\overline{G}$. Hence, for any edge $e \in E'$, its radius- ℓ neighborhood in $\overline{G}$ is contained entirely within $\mathcal{G}$ iff for every vertex $(v, i) \in V'$ which is at a distance of at most $\ell - 1$ from at least one of the endpoints of e satisfies $\deg_{\overline{G}}(v, i) = d(v, i)$. It follows that the a_T defined in the post-processing steps of Algorithm 2 is the same as the one in Theorem 3.9. Thus, conditioned on $\overline{G}$ satisfying the properties that we stated above and $2m \geq n_0$, with probability at least 9/10,

$$\text{tvd}(\mathcal{D}, \text{EdgeNbhdTypeDist}_{\overline{G};h}^{\ell}) \leq \epsilon/8.$$

Set $n_0 = \max \left\{ n_0^{(1-\epsilon^{1/\epsilon})^{-1}}, \left(\frac{1}{\epsilon^2}\right)^{\frac{1}{\epsilon}} \frac{25}{\epsilon} \right\}$. Thus, whenever $n \geq n_0$, we have $m \geq n_0$.

We finally argue that the termination conditions in Algorithm 2 happen with very low probability. Since $m \leq n^{1+\epsilon^{25/\epsilon}}$ and $|\overline{V}| = 2m$, we have $\mathbb{E}[V'] \leq n^{1-7\beta/8}$. By applying Chernoff bound, we conclude that with very high probability, $|V'| \leq n^{1-3\beta/4}$. For an edge $e \in \overline{E}$, let Z_e be the indicator random variable for the event that $e \in E'$. Let $Z = \sum_{e \in \overline{E}} Z_e$. We have $\mathbb{E}[Z_e] = n^{-2\beta} = n^{-2\epsilon^{20/\epsilon}}$ and $\text{Var}[Z] \leq (3520/\epsilon^2)\mathbb{E}[Z]$, since the maximum degree of any vertex in $\overline{G}$ is at most $3520/\epsilon^2$. Since $|\overline{E}| \leq 320m/\epsilon^2$, $\mathbb{E}[Z] \leq n^{1-\epsilon^{22/\epsilon}}$. By applying Chebyshev's inequality, we conclude that with high probability, $|E'| \leq n^{1-\epsilon^{25/\epsilon}}$. This concludes the proof of correctness of Algorithm 2. $\blacktriangleleft$

6 Two-pass streaming algorithm for Max-DICUT

In this section, we modify⁹ Algorithm 2 to give a two-pass streaming algorithm that achieves $(1/2 - \epsilon)$ -approximation on any n -vertex graph with average degree *at most* n^δ , using space $O(n^{1-\delta})$, for some $\delta \in (0, 1)$. Combining our algorithm with the two-pass algorithm of Bhaskara, Daruki, and Venkatasubramanian [5] (see Theorem 2.3) that achieves for every $\epsilon, \delta \in (0, 1)$, $(1 - \epsilon)$ -approximation for graphs of average degree *at least* n^δ using space $\tilde{O}(n^{1-\delta})$, we obtain our algorithm that works on arbitrary graphs (see Algorithm 4).

Algorithm 3 Two-pass streaming algorithm for Max-DICUT on graphs of bounded average degree.

Parameters: $n \in \mathbb{N}$, $\epsilon \in (0, 1/2)$

Input: A stream σ of edges of directed graph $G(V, E)$ on n vertices and at most $n^{1+\delta}$ edges, where $\delta = \epsilon^{25/\epsilon}$

⁹ The specific modifications are highlighted in the code.

First pass:

```

1: Maintain a counter for the number of edges  $m$ 
2: Set a parameter  $\beta \triangleq \epsilon^{20/\epsilon}$ 
3: Initialize  $V', E' = \emptyset$ 
4: Initialize an  $n$ -length array count (indexed by  $V$ ) to be  $0^n$ 
5: Initialize an  $n$ -length array est-deg (indexed by  $V$ ) to be  $0^n$ 
6: for every edge  $u \rightarrow v$  in the stream do
7:   Sample  $z_1, z_2 \sim \text{Bern}(n^{-\beta})$  independently
8:   if  $z_1 = 1$  then
9:     Add  $(u, \text{count}[u])$  to  $V'$  and increment  $\text{count}[u]$  by 1
10:  if  $z_2 = 1$  then
11:    Add  $(v, \text{count}[v])$  to  $V'$  and increment  $\text{count}[v]$  by 1
12:  if  $|V'| > n^{1-3\beta/4}$  then
13:    Terminate the execution
14:  Sample  $y_1, y_2 \sim \text{Bern}(n^{-\beta/4})$  independently
15:  if  $y_1 = 1$  then
16:    Increment est-deg $[u]$  by  $n^{\beta/4}$ 
17:  if  $y_2 = 1$  then
18:    Increment est-deg $[v]$  by  $n^{\beta/4}$ 
19:  if # non-zero entries in est-deg  $> n^{1-\beta/8}$  then
20:    Terminate the execution

21: for every  $v \in V$  such that  $\text{count}[v] > 0$  do
22:    $\forall i \in [\text{count}[v]]$ , initialize  $d(v, i) = 0$ 
23: Initialize an  $n$ -length array store-deg (indexed by  $v$ ) to be  $0^n$ 
24: Initialize  $\hat{E} = \emptyset$ 

```

Second pass:

```

25: if  $m \leq n^{1-\epsilon^{1/\epsilon}}$  then
26:   Store all the edges and compute the Max-DICUT value.
27: else
28:   Set a parameter  $d = 320/\epsilon^2$ 
29:   for every edge  $u \rightarrow v$  in the stream do
30:     if  $\text{count}[u] > 0$  and  $\text{store-deg}[u] < n^{2\beta/3}$  then
31:       Add  $u \rightarrow v$  to  $\hat{E}$ 
32:       Increment store-deg $[u]$ 
33:     if  $\text{count}[v] > 0$  and  $\text{store-deg}[v] < n^{2\beta/3}$  then
34:       Add edge  $u \rightarrow v$  to  $\hat{E}$  if it was not previously added in Algorithm 3
35:       Increment store-deg $[v]$ 

36:   Initialize  $i = 1$ 
37:   for  $i \leq d$  do
38:     if  $\text{count}[u] > 0$  and est-deg $(u) > 0$  then
39:       Sample  $i_1 \sim \text{Unif}([\text{est-deg}(u)])$ 
40:       if  $i_1 < \text{count}[u]$  then
41:         Increment  $d(u, i_1)$  by 1
42:     if  $\text{count}[v] > 0$  and est-deg $(v) > 0$  then

```

```

43: | | | Sample  $i_2 \sim \text{Unif}([\text{est-deg}(v)])$ 
44: | | | if  $i_2 < \text{count}[v]$  then
45: | | | | Increment  $d(v, i_2)$  by 1
46: | | | if  $(\text{count}[u] > 0, \text{est-deg}(u) > 0, i_1 < \text{count}[u])$  or  $(\text{count}[v] > 0,$ 
| | | |  $\text{est-deg}(v) > 0, i_2 < \text{count}[v])$  then
47: | | | | Add  $(u, i_1) \rightarrow (v, i_2)$  to  $E'$ 
48: | | | | if  $|E'| > n^{1-\epsilon^{25/\epsilon}}$  then
49: | | | | | Terminate the execution
50: | | | Increment  $i$  by 1

51: Post-processing I: Resampling edges incident to low-degree vertices:
52: Initialize  $E'' = E'$ 
53: for every  $u \in V$  such that  $\text{count}[u] > 0$  and  $\text{store-deg}[u] < n^{2\beta/3}$  do
54: | Initialize  $i = 0$ 
55: | for  $i \in [\text{est-deg}[u]]$  do
56: | | for every edge  $(u, i) \rightarrow (v, j) \in E''$  or  $(v, j) \rightarrow (u, i) \in E''$  do
57: | | | Delete the edge; decrement  $d(u, i)$  (resp.  $d(v, j)$ ) if  $(u, i) \in V'$  (resp.  $(v, j) \in V'$ )

58: for every  $u \rightarrow v \in \hat{E}$  do
59: | Initialize  $j = 1$ 
60: | for  $j \leq d$  do
61: | | if  $\text{count}[u] > 0$  and  $\text{store-deg}(u) < n^{2\beta/3}$  then
62: | | | Sample  $j_1 \sim \text{Unif}([\text{store-deg}(u)])$ 
63: | | | if  $j_1 < \text{count}[u]$  then
64: | | | | Increment  $d(u, j_1)$  by 1
65: | | | if  $\text{count}[v] > 0$  then
66: | | | | if  $\text{store-deg}(v) < n^{2\beta/3}$  then
67: | | | | | Sample  $j_2 \sim \text{Unif}([\text{store-deg}(v)])$ 
68: | | | | else
69: | | | | | Sample  $j_2 \sim \text{Unif}([\text{est-deg}(v)])$ 
70: | | | | if  $j_2 < \text{count}[v]$  then
71: | | | | | Increment  $d(v, j_2)$  by 1
72: | | | | | if  $j_1 < \text{count}[u]$  then
73: | | | | | | Add  $(u, j_1) \rightarrow (v, j_2)$  to  $E''$ 
74: | | else if  $\text{count}[v] > 0$  and  $\text{store-deg}(v) < n^{2\beta/3}$  then
75: | | | Sample  $j_2 \sim \text{Unif}([\text{store-deg}(v)])$ 
76: | | | if  $j_2 < \text{count}[v]$  then
77: | | | | Increment  $d(v, j_2)$  by 1
78: | | | if  $\text{count}[u] > 0$  then
79: | | | | Sample  $j_1 \sim \text{Unif}([\text{est-deg}(u)])$ 
80: | | | | if  $j_1 < \text{count}[u]$  then
81: | | | | | Increment  $d(u, j_1)$  by 1
82: | | | | | if  $j_2 < \text{count}[v]$  then
83: | | | | | | Add  $(u, j_1) \rightarrow (v, j_2)$  to  $E''$ 
84: | | | Increment  $j$  by 1
85: Delete every edge in  $E''$  that is incident on a vertex not in  $V'$ 

```

Post-processing II (analogous to the post processing steps in Algorithm 2)

1. For every $(u, i) \in V'$ such that $d(u, i) > 11d = 3520/\epsilon^2$, delete every edge incident on (u, i) in E'' and decrement $d(\cdot)$ each of its endpoints by 1. Finally, set $d(u, i) = 0$. Let $\mathcal{G}(V', E'')$ be the directed graph on vertex set V' and edge set E'' .
2. Let $\ell, c \in \mathbb{N}$ be set according to Corollary 3.7 for parameter $\epsilon'' = \epsilon/8$. Let $D = 11d$. Let $\mathcal{H}_{2m}$ be the hash family in Corollary 3.7 corresponding to directed graphs on $2m$ vertices. Sample $h \sim \text{Unif}(\mathcal{H}_{2m})$.
3. For $T \in \text{Typ}_c^{\ell, D}$, let a_T denote the number of edges $e \in E''$ such that every vertex $(v, i) \in V'$ which is at a distance of at most $\ell - 1$ from at least one of the endpoints of e satisfies the property that $\deg_{\mathcal{G}}(v, i) = d(v, i)$, and $\text{nbhdtype}_{\mathcal{G}, h|_{V'}}^{\ell}(e) = T$.
4. Consider the distribution $\mathcal{D} \in \Delta(\text{Typ}_c^{\ell, D})$, defined as

$$\mathcal{D}(T) = \frac{a_T \cdot n^{\beta|T|}}{\sum_{T' \in \text{Typ}_c^{\ell, D}} a_{T'} \cdot n^{\beta|T'|}},$$

where $T \in \text{Typ}_c^{\ell, D}$, and $|T|$ denotes the number of vertices in T .

5. Let $\text{Local} : \text{Typ}_c^{\ell, D} \rightarrow [0, 1]$ be the function in Corollary 3.7. Output $\mathbb{E}_{T \sim \mathcal{D}}[\text{Local}(T)]$.

► **Theorem 6.1.** *For every $\epsilon \in (0, 1/2)$, there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and every directed graph G on n vertices and $m \leq n^{1+\epsilon^{25/\epsilon}}$ edges, Algorithm 3 outputs a value v satisfying*

$$\frac{\text{Max-DICUT}(G)}{2} - \epsilon \leq v \leq \text{Max-DICUT}(G) + \epsilon,$$

with probability at least $2/3$, using space $n^{1-\epsilon^{O(1/\epsilon)}}$.

Proof. At a high level, the main change from Algorithm 2 is that for vertices randomly sampled from $\bar{V}$ in the first-pass, we don't use an additional pass like in Algorithm 2 to exactly compute the degrees of all their parent vertices. Instead, in the first pass, we approximately estimate the degrees of all high-degree vertices (of degree at least $n^{2\beta/3}$) with high probability. As shown in Theorem 4.1, for high-degree vertices, an approximate estimate of their degrees suffices for the reduction to work. In the second pass, we handle the low degree vertices and store all the edges incident to parent vertices of degree less than $n^{2\beta/3}$ and resample their copies in $\bar{G}$ in the post-processing steps from Algorithm 3 to Algorithm 3. In addition, for technical reasons, E' now stores *all* edges incident on vertices in V' , not just the induced edges (see Algorithm 3). This helps us maintain accurate degrees of the sampled vertices when we delete the edges sampled in the second pass that were incident to copies of low degree vertices and resample them later in post-processing I. We now formally analyze the space usage and argue the correctness of Algorithm 3.

Space usage of Algorithm 3. Compared to Algorithm 2, Algorithm 3 also stores **est-deg**, **store-deg**, and $\hat{E}$. It follows from the termination condition in Algorithm 3 that **est-deg** has at most $n^{1-\beta/8}$ non-zero entries, and the value stored in each entry is at most $\text{poly}(n)$. Thus, the space required to store **est-deg** is at most $n^{1-\epsilon^{O(1/\epsilon)}}$. The termination condition in Algorithm 3 ensures that $|V'| \leq n^{1-3\beta/4}$. The number of non-zero entries in **store-deg** is at most $|V'|$. Finally, whenever an edge is added to $\hat{E}$, both its endpoints have copies in V' and at least one of them currently has fewer than $n^{2\beta/3}$ edges incident on it in $\hat{E}$. Therefore, the number of edges stored in $\hat{E}$ is at most $|V'|n^{2\beta/3} \leq n^{1-\beta/12}$.

Correctness of Algorithm 3. For now, let us ignore the termination conditions in Algorithm 3. We will later argue that these events occur with very low probability, and hence, do not alter the success probability of Algorithm 3 by much. We first prove the following claim.

▷ **Claim 6.2.** Let n be sufficiently large. With probability at least $1 - \exp(-n^{-\Omega(\epsilon^{O(1/\epsilon)})})$, for every $v \in V$ such that $\deg(v) \geq n^{2\beta/3}$, we have

$$(1 - \epsilon/100) \deg(v) \leq \text{est-deg}[v] \leq (1 + \epsilon/100) \deg(v).$$

Proof. Fix any vertex $v \in V$ whose degree is at least $n^{2\beta/3}$. For $i \in [\deg(v)]$, let $Y_i \sim \text{Bern}(n^{-\beta/4})$ independently. Then, $\text{est-deg}[v] = \sum_{i \in [\deg(v)]} Y_i \cdot n^{\beta/4}$. Therefore, $\mathbb{E}[\text{est-deg}[v]] = \deg(v)$. By applying Hoeffding's inequality (Theorem 3.3), we conclude that with probability at least $1 - \exp(-n^{-\Omega(\epsilon^{O(1/\epsilon)})})$,

$$(1 - \epsilon/100) \deg(v) \leq \text{est-deg}[v] \leq (1 + \epsilon/100) \deg(v).$$

The claim follows from taking a union bound over at most n vertices. $\triangleleft$

Now let us condition on **est-deg** that satisfies the above property. We define an n -length array **approx-deg** as follows. For every vertex $v \in V$, if $\deg(v) < n^{2\beta/3}$, then **approx-deg** $[v] := \deg(v)$, else **approx-deg** $[v] := \text{est-deg}[v]$. Consider the bounded-degree graph $\bar{G}(\bar{V}, \bar{E})$ ¹⁰ that is output by Algorithm 1, corresponding to the input graph $G(V, E)$, parameters $\epsilon' = \epsilon/2$ and $\zeta = 2\beta/3$, and **approx-deg**. It follows from Theorem 4.1 that with probability at least $5/6$,

- $\text{Max-DICUT}(G) - \epsilon/2 \leq \text{Max-DICUT}(\bar{G}) \leq \text{Max-DICUT}(G) + \epsilon/2$,
- the maximum degree of any vertex in $\bar{G}$ is at most $110/(\epsilon')^2 = 3520/\epsilon^2$, and
- $\bar{G}$ has exactly $2m$ vertices and at least $240m/\epsilon^2$ edges.

We will prove the following claim (analogous to Claim 5.2 in Theorem 5.1).

▷ **Claim 6.3.** At the end of post-processing I, the following hold:

1. V' has the same distribution as sampling every vertex from $\bar{V}$ independently with probability $n^{-\beta}$, and applying a relabeling on the vertices, and
2. for $u \rightarrow v \in E$, if both u and v have copies in V' , then the copies of $u \rightarrow v$ in $\tilde{G}$ are sampled identically in Algorithms 1 and 3. If only one of u or v has a copy in V' , then the endpoints corresponding to the copies of that vertex in the copies of $u \rightarrow v$ in $\tilde{G}$ are sampled identically in Algorithms 1 and 3.

Proof. The proof of Item 1 of Claim 6.3 is analogous to the proof of Item 1 of Claim 5.2. To prove Item 2, we first consider the edges incident to high-degree vertices which have copies in V' . Since **approx-deg** and **est-deg** are the same for vertices of degree at least $n^{2\beta/3}$, the sampling in Algorithm 3 of Algorithm 3 is the same as the sampling in Algorithm 1 of Algorithm 1 for edges whose both endpoints are of degree at least $n^{2\beta/3}$ in G . Similar, if only one of the endpoints of $u \rightarrow v$ has a copy in V' and if the parent vertex of that copy is of degree at least $n^{2\beta/3}$, then the endpoints corresponding to the copies of that vertex in the copies of $u \rightarrow v$ in $\tilde{G}$ are sampled identically in Algorithms 1 and 3. Now consider the edges incident to low-degree vertices which have copies in V' . While these edges are **not** sampled with the correct probabilities in Algorithm 3, this is fixed in the post processing steps

¹⁰We implicitly assume that Algorithm 1 uses the same randomness as in Algorithm 3 to sample copies of the edges incident on the parent vertices of vertices in V' . To sample the remaining edges, it uses independent random bits.

from Algorithm 3 to Algorithm 3. For every $(u, i) \in V'$, observe that $\text{store-deg}[u] < n^{2\beta/3}$ if and only if $\deg(u) < n^{2\beta/3}$. In addition, when $\deg(u) < n^{2\beta/3}$, $\text{store-deg}[u] = \deg[u]$. In post-processing I, for every low-degree vertex that has a copy in V' , we delete all edges in E'' that are incident on *any* of its copies, including those not contained in V' . All edges incident to low-degree vertices (which have a copy in V') are stored in $\hat{E}$. Since $\text{store-deg}[u] = \deg[u]$ for these vertices, we resample the copies of the edges incident on them identical to the sampling in Algorithm 1. $\triangleleft$

Given the above claim, we consider $\tilde{G}$ where for $u \rightarrow v \in E$, if both u and v have copies in V' , then the copies of $u \rightarrow v$ are sampled according to Algorithm 3 (prior to post-processing II). If only one of u or v has a copy in V' , then the endpoints corresponding to the copies of that vertex in the copies of $u \rightarrow v$ are sampled according to Algorithm 3 and the other endpoints are sampled independently, according to Algorithm 1. The remaining edges are also sampled according to Algorithm 1. Finally, like in Algorithm 1, we define $\bar{G}$ as $\tilde{G}$ with edges incident on vertices of degree larger than $11d$ deleted. Observe that for $\bar{G}$ as defined above, $\mathcal{G}(V', E'')$ is the induced subgraph of $\bar{G}$ on V' . Furthermore, at the end of post-processing II of Algorithm 3, for $(v, i) \in V'$, $d(v, i)$ is degree of (v, i) in $\bar{G}$. Now, the rest of the proof of correctness follows similarly as in the proof of Theorem 5.1.

Finally, we argue that the termination conditions in Algorithm 3 happen with very low probability. Observe that V' is sampled exactly in the same way as in Algorithm 2. Therefore, the same proof goes through. To analyze the number of non-zero entries in est-deg , for $i \in [\log n]$, consider any vertex $v \in V$ whose degree is larger than 2^i and at most 2^{i+1} . Let Z_v^i be the indicator variable for the event that $\text{est-deg}[v] \neq 0$ and let $Z^i = \sum_{v: 2^i < \deg(v) \leq 2^{i+1}} Z_v^i$. Observe that $\mathbb{E}[Z_v^i] \leq 2^{i+1}n^{-\beta/4}$ and by the linearity of expectation,

$$\mathbb{E}[Z^i] \leq \frac{2m}{2^i} \cdot 2^{i+1}n^{-\beta/4} = 4mn^{-\beta/4} \leq n^{1-\beta/6}.$$

Since the events are independent, applying Chernoff bound (Theorem 3.2), we conclude that with probability at least $1 - \exp(-\Omega(\epsilon^{O(1/\epsilon)}))$, $Z^i \leq (\log n)n^{1-\beta/6}$. Taking a union bound over all $i \in [\log n]$, we get that with high probability, $\sum_{i \in [\log n]} Z^i \leq (\log n)^2 n^{1-\beta/6} < n^{1-\beta/8}$. We now analyze the termination condition on E' in Algorithm 3. Let E'_ℓ denote the set of edges in E' which are incident to vertices in V' whose parent vertices have degree less than $n^{2\beta/3}$. Let E'_h denote the set of edges in E' which are incident to vertices in V' whose parent vertices have degree at least $n^{2\beta/3}$. Observe that $E' = E'_\ell \cup E'_h$. We previously argued (in Section 5) that with high probability, $|V'| \leq n^{1-3\beta/4}$. Therefore, with high probability, $|E'_\ell| \leq n^{1-\beta/12}$. It follows from Claim 6.2 that with high probability, for every $v \in V$ such that $\deg(v) \geq n^{2\beta/3}$, we have

$$(1 - \epsilon/100) \deg(v) \leq \text{est-deg}[v] \leq (1 + \epsilon/100) \deg(v).$$

Therefore, conditioned on this event, $\mathbb{E}[|E'_h|] \leq dn^{1+\delta}(2n^{-\beta}(1 - \epsilon/100)^{-1}) \leq n^{1-3\beta/4}$, for sufficiently large n . Applying Markov's inequality, we conclude that with high probability, $|E'_h| \leq n^{1-2\beta/3}$. Thus, the termination condition on E' in Algorithm 3 occurs with very low probability, and this concludes the proof of correctness of Algorithm 3. $\blacktriangleleft$

Below, we describe our algorithm for Max-DICUT on arbitrary graphs.

Algorithm 4 Two-pass streaming algorithm for Max-DICUT.

Parameters: $n \in \mathbb{N}$, $\epsilon > 0$

Input: A stream σ of edges of a directed graph $G(V, E)$ on n vertices

First pass:

- 1: Maintain a counter for the number of edges m and execute the first passes of Algorithm 3 and the algorithm in [5], in parallel

Second pass:

- 2: **if** $m \leq n^{1+\epsilon^{4/\epsilon}}$ **then**
 - 3: Continue executing the second pass of Algorithm 3 and abandon the execution of [5]
 - 4: **else**
 - 5: Abandon the execution of Algorithm 3 and execute the second pass of [5]
-

Finally, we conclude with a proof of our main theorem.

Proof of Theorem 1.1. The proof of correctness and the space usage of Algorithm 4 immediately follows from Theorem 6.1 and Theorem 2.3. $\blacktriangleleft$

References

- 1 Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part II, ICALP '09*, pages 328–338, Berlin, Heidelberg, 2009. Springer-Verlag. doi:10.1007/978-3-642-02930-1_27.
- 2 Noga Alon, Wenceslas Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csps. *J. Comput. Syst. Sci.*, 67(2):212–243, 2003. doi:10.1016/S0022-0000(03)00008-4.
- 3 Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21–25, 2021*, pages 612–625. ACM, 2021. doi:10.1145/3406325.3451110.
- 4 Boaz Barak, Moritz Hardt, Thomas Holenstein, and David Steurer. Subsampling mathematical relaxations and average-case complexity. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23–25, 2011*, pages 512–531. SIAM, 2011. doi:10.1137/1.9781611973082.41.
- 5 Aditya Bhaskara, Samira Daruki, and Suresh Venkatasubramanian. Sublinear algorithms for MAXCUT and correlation clustering. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9–13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.16.
- 6 Keren Censor-Hillel, Rina Levy, and Hadas Shachnai. Fast distributed approximation for max-cut. In Antonio Fernández Anta, Tomasz Jurdzinski, Miguel A. Mosteiro, and Yanyong Zhang, editors, *Algorithms for Sensor Systems - 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2017, Vienna, Austria, September 7–8, 2017, Revised Selected Papers*, volume 10718 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2017. doi:10.1007/978-3-319-72751-6_4.

- 7 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, Ameya Velingker, and Santhoshini Velusamy. Linear space streaming lower bounds for approximating csps. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 275–288. ACM, 2022. doi:10.1145/3519935.3519983.
- 8 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all boolean csps with linear sketches, 2022. arXiv:2102.12351.
- 9 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Sketching approximability of all finite csps. *J. ACM*, 71(2):15:1–15:74, 2024. doi:10.1145/3649435.
- 10 Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal streaming approximations for all boolean max-2csps and max-ksat. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 330–341. IEEE, 2020. doi:10.1109/FOCS46700.2020.00039.
- 11 Yumou Fei, Dor Minzer, and Shuo Wang. Multi-pass streaming lower bounds for approximating max-cut. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*, pages 1537–1560. IEEE, 2025. doi:10.1109/FOCS63196.2025.00081.
- 12 Yumou Fei, Dor Minzer, and Shuo Wang. A dichotomy theorem for multi-pass streaming csps. In Aditya Bhaskara and Artur Czumaj, editors, *Proceedings of the 58th Annual ACM Symposium on Theory of Computing, STOC 2026, Salt Lake City, UT, USA, June 22-26, 2026*, pages 246–255. ACM, 2026. doi:10.1145/3798129.3800744.
- 13 Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for MAX CUT. *Random Struct. Algorithms*, 20(3):403–440, 2002. doi:10.1002/RSA.10036.
- 14 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. doi:10.1145/227683.227684.
- 15 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.
- 16 Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. Streaming complexity of approximating max 2csp and max acyclic subgraph. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 8:1–8:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.8.
- 17 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1263–1282. SIAM, 2015. doi:10.1137/1.9781611973730.84.
- 18 Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. $(1 + \Omega(1))$ -approximation to MAX-CUT requires linear space. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1703–1722. SIAM, 2017. doi:10.1137/1.9781611974782.112.
- 19 Michael Kapralov and Dmitry Krachun. An optimal space lower bound for approximating MAX-CUT. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 277–288. ACM, 2019. doi:10.1145/3313276.3316364.
- 20 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

- 21 Claire Mathieu and Warren Schudy. Yet another algorithm for dense max cut: go greedy. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 176–182. SIAM, 2008. URL: <http://dl.acm.org/citation.cfm?id=1347082.1347102>.
- 22 Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 245–254. ACM, 2008. doi:10.1145/1374376.1374414.
- 23 Raghuvansh R. Saxena, Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming complexity of csps with randomly ordered constraints. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 4083–4103. SIAM, 2023. doi:10.1137/1.9781611977554.CH156.
- 24 Raghuvansh R. Saxena, Noah G. Singer, Madhu Sudan, and Santhoshini Velusamy. Improved streaming algorithms for maximum directed cut via smoothed snapshots. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 855–870. IEEE, 2023. doi:10.1109/FOCS57990.2023.00055.
- 25 Raghuvansh R. Saxena, Noah G. Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming algorithms via local algorithms for maximum directed cut. In Yossi Azar and Debmalaya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 3392–3408. SIAM, 2025. doi:10.1137/1.9781611978322.111.
- 26 Noah G. Singer, Madhur Tulsiani, and Santhoshini Velusamy. Sketching approximations and lp approximations for finite csps are related, 2025. doi:10.48550/arXiv.2509.17926.
- 27 Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 453–461. ACM, 2001. doi:10.1145/380752.380839.
- 28 Luca Trevisan. Max cut and the smallest eigenvalue. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 263–272. ACM, 2009. doi:10.1145/1536414.1536452.
- 29 Yuichi Yoshida. Optimal constant-time approximation algorithms and (unconditional) in-approximability results for every bounded-degree CSP. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, pages 665–674. Association for Computing Machinery, 2011. doi:10.1145/1993636.1993725.