Streaming Algorithms via Local Algorithms for Maximum Directed Cut*

Raghuvansh R. Saxena† Noah G. Singer‡ Madhu Sudan§ Santhoshini Velusamy¶

Abstract

We explore the use of local algorithms in the design of streaming algorithms for the Maximum Directed Cut problem. Specifically, building on the local algorithm of (Buchbinder, Feldman, Seffi, and Schwartz [14] and Censor-Hillel, Levy, and Shachnai [16]), we develop streaming algorithms for both adversarially and randomly ordered streams that approximate the value of maximum directed cut in bounded-degree graphs. In n-vertex graphs, for adversarially ordered streams, our algorithm uses $O(n^{1-\Omega(1)})$ (sub-linear) space and for randomly ordered streams, our algorithm uses logarithmic space. Moreover, both algorithms require only one pass over the input stream. With a constant number of passes, we give a logarithmic-space algorithm which works even on graphs with unbounded degree on adversarially ordered streams. Our algorithms achieve any fixed constant approximation factor less than 1/2. In the single-pass setting, this is tight: known lower bounds show that obtaining any constant approximation factor greater than 1/2 is impossible without using linear space in adversarially ordered streams (Kapralov and Krachun [37]) and $\Omega(\sqrt{n})$ space in randomly ordered streams, even on bounded degree graphs (Kapralov, Khanna, and Sudan [35]).

In terms of techniques, our algorithms partition the vertices into a small number of different types based on the structure of their local neighborhood, ensuring that each type carries enough information about the structure to approximately simulate the local algorithm on a vertex with that type. We then develop tools to accurately estimate the frequency of each type. This allows us to simulate an execution of the local algorithm on all vertices, and thereby approximate the value of the maximum directed cut.

1 Introduction

We give almost 1/2-approximation algorithms solving the *maximum directed cut* (Max-DICUT) problem in graphs in a variety of streaming settings, by appealing to local algorithms achieving similar approximations. We describe the problem and settings in more detail below before turning to the results and techniques.

1.1 The Max-DICUT Problem and its Significance

The *maximum directed cut* problem (Max-DICUT) is the problem of estimating the value of the maximum directed cut in an input graph G. Here, a (directed) cut is a subset S of the vertices and the value of the cut is the fraction of edges (u, v) in the graph satisfying $u \in S$ and $v \notin S$. We denote the value of the largest cut by maxval_G and say that an algorithm produces an α-approximation if it guarantees to output a value at least $\alpha \cdot \mathsf{maxval}_G$ (and at most maxval_G) on all graphs G.

Besides being a central problem in its own right, Max-DICUT is also significant as it is an example of a *Constraint Satisfaction Problem (CSP)*. In a constraint satisfaction problem, there is a set of variables and a set of constraints over these variables, and the goal is find out the maximum number of constraints that can be satisfied by an assignment to the variables. CSPs form an infinite set of problems that often capture many natural settings, and have received considerable attention in both streaming [39, 35, 29, 36, 10, 37, 4, 5, 21, 11, 17, 20, 21, 32, 50, 49, 40, 53, 52, 33] and non-streaming settings [51, 26, 6, 48, 7, 38, 45, 15, 55, among many others]. For streaming settings, the Max-DICUT problem has emerged as a leader on the algorithmic front, with almost all algorithms being developed first for the Max-DICUT problem before being extended to other CSPs. In fact, speaking in broad strokes, CSPs form an infinite class of problems, in almost all contexts they tend to have a finite classification and in particular there is a finite set of algorithms that essentially cover the entire class. Thus any algorithm that works well for any CSP problem offers hope for the entire class; and the Max-DICUT problem has proven to be the most suitable for algorithmic developments and insights.

*The full version of the paper is forthcoming on arXiv.

†Tata Institute of Fundamental Research, Mumbai, Maharashtra, India. Email: raghuvansh.saxena@gmail.com.

‡Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. Email: ngsinger@cs.cmu.edu.

§School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts, USA. Email: madhu@cs.harvard.edu.

¶Toyota Technological Institute, Chicago, Illinois, USA. Email: santhoshini@ttic.edu.

Citation	Approx. factor	Input order	Space	Passes	Bounded-degree?
Folklore	$1-\epsilon$	Adversarial	$\widetilde{O}(n)$	1	No
[29]	$2/5-\epsilon$	Adversarial	$O(\log n)$	1	No
[23]	$4/9-\epsilon$	Adversarial	$O(\log n)$	1	No
[50]	0.485	Adversarial	$\widetilde{O}(\sqrt{n})$	1	Yes
[50]	0.485	Random	$O(\log n)$	1	No
[50]	0.485	Adversarial	$O(\log n)$	2	No
[49]	0.485	Adversarial	$\widetilde{O}(\sqrt{n})$	1	No
This work	$1/2-\epsilon$	Adversarial	$o(n)$	1	Yes
This work	$1/2-\epsilon$	Random	$O(\log n)$	1	Yes
This work	$1/2-\epsilon$	Adversarial	$O(\log n)$	$O(1)$	No

Table 1: A table of known streaming algorithms for Max-DICUT. Some lower bounds (that hold even for bounded degree graphs) are also known: $(4/9+\epsilon)$-approximation in single-pass adversarial-ordering streams requires $\Omega(\sqrt{n})$ space [23]; $(1/2+\epsilon)$-approximation in single-pass adversarial-ordering streams requires $\Omega(n)$ space [37]; $(1/2+\epsilon)$-approximation in single-pass random-ordering streaming requires $\Omega(\sqrt{n})$ space [35]; and $(1-\epsilon)$-approximation over adversarially-ordered streams requires either $n^{\Omega(1)}$ space or $\Omega(1/\epsilon)$ passes [5]. We conjecture that the $(1/2+\epsilon)$ lower bounds can be simultaneously generalized, to show that $(1/2+\epsilon)$-approximation in single-pass random-ordering streaming requires $\Omega(n)$ space.

1.2 Streaming Algorithms for Max-DICUT In the streaming model, the input graph G is presented to the algorithm as a stream of edges, whose goal is to make one or passes over this stream and produce an α-approximation using as little memory as possible. (In this paper, all algorithms are allowed to toss random coins and need to succeed with probability 99% over the choice of random coins for every input.) It is easy to see that storing $O(n)$ randomly chosen edges suffices to produce a $(1-\epsilon)$-approximation, and this can be done in a single pass and $\widetilde{O}(n)$ memory. However, a linear-sized memory is very often unaffordable and thus, research has mostly focused on investigating the power of sublinear algorithms for Max-DICUT.

On this front, the work of [37] showed that $\Omega(n)$ space is needed to achieve any approximation better than $1/2$ when the edges are ordered adversarially, even on bounded degree graphs. It is believed that the proof techniques also extend to randomly ordered streams although the best bound known is a memory lower bound of $\Omega(\sqrt{n})$ in the early work of [35]. Thus, the best approximation factor one can hope for with sublinear space is $1/2$ and here, the Max-DICUT problem does admit many non-trivial results.

The first non-trivial streaming algorithm for approximating Max-DICUT was due to [29] who gave a single-pass that uses logarithmic space and produces a $2/5$ approximation. Subsequently, [23] improved this to a $4/9$ approximation and also proved that a better approximation is impossible without using at least $\Omega(\sqrt{n})$ space if the edges are ordered adversarially. Subsequently, [50, 49] showed that even better approximations (of around 0.485) can be produced if algorithms are allowed $\widetilde{O}(\sqrt{n})$ space. Moreover, the space bound improves to logarithmic if the edges in the graph as assumed to arrive in a random order.

The above works, summarized in Table 1, have led to improvements for a wide class of constraint satisfaction problems. For example, [22] extended the work of [23] mentioned above to cover all possible CSPs and [53] showed that the algorithms in [50, 49] can also be extended to a richer class. This again demonstrates the significance of algorithms for Max-DICUT when it comes to designing algorithms for general CSPs. However, the following question remains open: Can sublinear space streaming algorithms achieve $(1/2-\epsilon)$-approximations for the Max-DICUT problem, for *every* $\epsilon > 0$? While we do not resolve this question here, we show many special cases, in particular including most used/sufficient for existing lower bounds, do have such an approximation algorithm in sublinear space.

1.3 Our Results As mentioned above, we achieve optimal $(1/2-\epsilon)$-approximation for the Max-DICUT problem in various restricted settings.

Bounded degree graphs with adversarially ordered edges. The first variant we consider is when the input graph is assumed to have a bounded degree, i.e., the degree of any vertex is assumed to be at most a pre-specified constant. While the bounded degree setting does make it easier to effect algorithmic improvements,

it also tends to be a strong predictor of general results, in that the algorithms can often be extended (possibly with many complications in algorithm design as well as analysis) to the general setting. As an example, the $\widetilde{O}(\sqrt{n})$-space Max-DICUT algorithm of [50] for bounded-degree graphs was extended to general graphs in [49]. Conversely, as far as we are aware, all known streaming lower bounds for Max-DICUT (i.e., [35, 37]) are based on random sparse graphs where the maximum degree of any vertex is at most constant (or logarithmic). Together, these past algorithms and lower bounds motivate the search for algorithms in this setting. For this setting, we show that[1]:

THEOREM 1.1. (ADVERSARIAL-ORDER ALGORITHM FOR BOUNDED DEGREE GRAPHS) *For every $D \in \mathbb{N}$ and $\epsilon > 0$, there is a streaming algorithm which $(1/2 - \epsilon)$-approximates the* Max-DICUT *value of an n-vertex graph with maximum degree at most D in $O(n^{1-\Omega(1)})$ space using a single, adversarially-ordered pass over the list of edges.*

Bounded degree graphs with randomly ordered edges. The random order streaming model is now a central model in streaming literature, and understanding it is an important quest of its own [34, 44, 47, 25, 2, 9, 30, etc.]. The difference between the random order model and the adversarially ordered model is that in the random order model, the edges of the input graph[2] are presented in a uniformly random order to the streaming algorithm, and the algorithm has to do well on most orders. [50] show that it is possible to produce an approximation of around 0.485 of Max-DICUT using only logarithmic space, thereby going past the 4/9 lower bound that holds for adversarially ordered streams [23]. However, there is still a gap between the approximation guarantee of the best known algorithm (around 0.485) and the best known lower bound of $1/2 + \epsilon$ shown in [35], that also holds for bounded degree graphs. Our work here closes this gap for bounded degree graphs.

THEOREM 1.2. (RANDOM-ORDER ALGORITHM FOR BOUNDED DEGREE GRAPHS) *For every $D \in \mathbb{N}$ and $\epsilon > 0$, there is a streaming algorithm which $(1/2 - \epsilon)$-approximates the* Max-DICUT *value of an n-vertex graph with maximum degree at most D in $O(\log n)$ space using a single, randomly-ordered pass over the list of edges.*

Multi-pass algorithms. Finally, we consider the setting where the streaming algorithm is allowed to make multiples passes over the (adversarially ordered) input stream. The multi-pass setting has also been widely studied [27, 24, 41, 28, 31, 1, 8, 13, 3, 4, 19, 18, 17, 12] in the streaming literature. In the context of Max-DICUT, the most relevant work is that of [50] that produces an approximation of around 0.485 using only two passes and logarithmic space, thereby surpassing the 4/9 lower bound that holds for single-pass algorithms [23]. We show that a constant number of additional passes can push the approximation factor arbitrarily close to 1/2. It is believed that, for the related problem of Max-CUT (which implies it[3] for Max-DICUT), even algorithms with logarithmically many passes need polynomial space to produce an approximation larger that 1/2 [17].

THEOREM 1.3. (MULTI-PASS ALGORITHM) *For every $\epsilon > 0$, there is a streaming algorithm which $(1/2 - \epsilon)$-approximates the* Max-DICUT *value of an arbitrary n-vertex graph in $O(\log n)$ space using $O(1/\epsilon)$ adversarially-ordered passes over the list of edges.*

1.4 Our Techniques The main idea behind our results is to import the local algorithms of [14, 16] for Max-DICUT into the streaming paradigm to get $(1/2 - \epsilon)$-approximations. A similar theme of translating local algorithms to sublinear algorithms was explored in the work of [46] for problems like minimum vertex cover.

The Max-DICUT algorithm of [14, 16] is "local" in the following sense: each vertex gets a (fractional) assignment depending only on its radius-k neighborhood.[4] We can essentially assume without loss of generality that our input graphs are k-colored for some large constant k (using a random k-coloring and discarding improperly colored edges), and so our algorithms focus on the task of understanding k-neighborhoods in a k-colored graph.

Before going into the details, we mention a different approach to local $(1/2-\epsilon)$-approximations for Max-DICUT: Kuhn, Moscibroda, and Wattenhofer [43] developed a local algorithm for $(1 - \epsilon)$-approximating the value

[1]Throughout, the $O(\cdot)$ and $\Omega(\cdot)$ notations hide constants depending on ϵ and, if present, D.

[2]The input graph is still worst-case, just that its edges are presented in a random order.

[3]An algorithm for solving Max-DICUT implies an algorithm for Max-CUT, simply replace every edge by two edges, one in each direction.

[4]Or more properly, on the *isomorphism class* of this neighborhood, i.e., the algorithm does not depend on the "names" of vertices in the neighborhood.

of packing-and-covering linear programs (see also the thesis of Kuhn [42]), and there is a well-known linear programming relaxation of Max-DICUT (see e.g. [54]) that is a packing-and-covering LP and is half-integral. Combining them could give a local algorithm for $(1/2 - \epsilon)$-approximating Max-DICUT, although we do not investigate this and make no formal claim in this regard. Instead, we use the algorithm in [14, 16].

We now explain the high-level idea behind the algorithms in Theorems 1.1 to 1.3:

Bounded degree graphs with adversarially ordered edges (Theorem 1.1). The starting point of this algorithm is the observation that, in the bounded-degree setting, each vertex has only a constant number of other vertices in its constant-radius neighborhood. This in turn means that there are only a constant number of different (constant-radius) neighborhoods a vertex can have up to isomorphism. As simulating a constant-radius local algorithm on a given vertex only requires knowledge of this neighborhood, we can simulate the local algorithm on all vertices if we know the number of vertices with a given neighborhood.

To formalize this, we define the "type" of a vertex to be its k-neighborhood (for an appropriate radius k depending on ϵ) up to isomorphism and estimate the frequency of each type by subsampling. Specifically, we sample a set S of vertices of (sublinear) size $O(n^{1-\Omega(1)})$ and use our pass to record the induced subgraph on the vertices S and the in-degree and out-degree of each vertex in S. With an appropriate choice of the constants, it can be shown that this sampling procedure would sample the entire neighborhood of a vertex with a certain probability that depends on its type. Moreover, the degree information we measure lets us detect whenever we sample the entire neighborhood of a given vertex. Thus, to estimate the frequency of a type in the whole graph, we can simply count the number of times we saw that type in our sample and scale it appropriately. Similar (but less general) tools were also present in [50].

Before continuing to the other results, let us state a subtlety that we overlooked in our description above. To estimate the Max-DICUT using the [14, 16] algorithm, we not only need to know the frequency of each type, but also need to know how many vertices of each type are connected to how many vertices of another type. Thus, instead of considering neighborhoods of vertices, we actually need to consider neighborhoods of edges. In our actual algorithms, we extend the ideas above appropriately to make them work for edges as well.

Bounded degree graphs with randomly ordered edges (Theorem 1.2). This algorithm builds on the foregoing one. Just as before, we still assume that the maximum degree of the vertices is bounded, and we can still partition the vertices (and the edges) into a constant number of types and try to compute the frequency of each type. The main challenge is that we are now trying to do this in logarithmic (instead of simply sublinear) space using the fact that the edges are presented randomly[5].

This means that we can no longer sample a set of vertices of size $O(n^{1-\Omega(1)})$ and consider the induced subgraph. Instead, we sample logarithmically many vertices and "build" their neighborhoods as the stream passes. For example, if we are building the neighborhood of vertex i (say), we start looking for edges with vertex i and add them to the neighborhood. As soon as we find such an edge (i, j) (say), we additionally start looking (recursively) for edges with vertex j until we reach our fixed (constant) depth k. The hope is that we will explore the entire neighborhood of vertex i and thereby compute its type.

However, there is a significant issue with this approach: Since the graph's edges appear in a random order, we will often only build a strict subset of the neighborhood. For instance, suppose our graph has the edges $(1, 2), (1, 3), (3, 4)$, and we are building the neighborhood of vertex 1. If $(3, 4)$ occurs before $(1, 3)$ in the stream, it will "slip by" and we will not include it in the neighborhood. Moreover, there is no way to determine at the end of the stream whether or not edges have slipped by. Thus, any neighborhood in the original graph will give rise to a distribution of neighborhoods in the sample based on which edges slipped by our algorithm. This affects our counts and therefore, the output of the algorithm.

To get around this, we use the fact that the distribution of neighborhoods is determined solely by the order of the edges in the stream. As we know the stream is uniformly random, we get that each neighborhood in the original graph G gives rise to a fixed distribution of neighborhoods in our sample. The fact that these distributions are fixed means that we can use Bayes' rule to compute the frequency of neighborhoods in G from the frequency of neighborhood in our sample. We mention that similar tools for measuring distributions were also used in [44]. (In fact, [44] contains a black-box algorithm for estimating the neighborhood-type distribution of random vertices. But as mentioned above, our actual algorithm needs to understand neighborhoods of *edges*, not vertices. Our

[5]Recall that the [23] lower bound implies that getting logarithmic space is impossible if the edges are presented in an adversarial order.

main technical contribution here is thus a modification of the [44] algorithm to this setting.)

We remark that our single-pass streaming algorithms measure 'strong' information about the input graph: The distribution of the type of the induced subgraph on the union of the k-balls around u and v, for a random edge (u, v). This turns out to be the correct abstraction of "edge-type distribution" for these algorithms. For instance, in the adversarial-ordering algorithm, we need to know how many vertices are in the union of the balls around u and v in order to calculate the probability that they are all in the subsampled set S. (This probability is in turn needed for the final reweighting step.) However, we emphasize that the local algorithm only uses much 'weaker' information about a random edge (u, v); for instance, it does not need to know how the ball around u intersects with the ball on v, nor does it need to know about neighbors w of u or v whose color is larger than u's and v's colors.

Multi-pass algorithms (Theorem 1.3). If a graph is promised to be bounded degree, then there is a simple and deterministic way to measure the frequency of neighborhoods of vertices using logarithmic space and constantly many passes: Given a starting vertex v, use the first pass to query the neighbors of v; the second pass to query these neighbors' neighbors, and so on. This procedure is guaranteed to exactly compute the neighborhood of v and uses only logarithmic space if the graph has bounded-degree.

But our multi-pass algorithm avoids making a bounded-degree assumption about the input graph. Thus, its flavor is quite different from the previous two single-pass algorithms, because unlike those algorithms, we can no longer afford to store the entire neighborhoods of vertices, which may be arbitrarily large. In particular, we can no longer rely on the [14, 16] algorithm as a black box.

Recall that the [14, 16] algorithm uses the entire k-neighborhood of a vertex v to produce a fractional assignment $x_v \in [0, 1]$ for v. Informally, we "robustify" the [14, 16] algorithm to produce an estimated fractional assignment for v based on *random subsampling* of its k-neighborhood. We show that this estimate is likely close to x_v. The subsampled graph is bounded-degree (in fact, $O(1)$-regular) and therefore our algorithm uses only logarithmic space.

We now give a brief overview of the [14, 16] and our modification. To compute the (fractional) assignment $x_v \in [0, 1]$ for a vertex v in G, the [14, 16] algorithm only uses a few quantities:

1. The sum of fractional assignments x_v for *lower*-colored neighbors of v.
2. Simple degree statistics of v: how many in- and out-edges in G does v have to lower- and higher-colored neighbors. (In the base case, color-1 vertices, the assignment depends only on these statistics.)

Note that only the first item involves recursive applications of the algorithm. Also, the recursion only has depth k, since we recurse only on lower-colored neighbors and there are k total colors.

Our robust local version of the [14, 16] algorithm should be interpreted as a random truncation/pruning of that algorithm's recursion tree so that every vertex makes a constant number of recursive calls. Indeed, we *estimate* the sum in the first item above by randomly sampling a constant-sized subset of v's lower-color neighbors and only recursing on these neighbors.

To make this more precise, our robust local algorithm looks like the following. For every vertex $v \in G$, we recursively define a distribution Y_v. To sample from Y_v:

1. Sample D random and independent lower-color neighbors $u_1, \ldots, u_D$ of v, where D is a large constant depending on ϵ. (Note: For convenience, the neighbors $u_1, \ldots, u_D$ are sampled with replacement. So even if v has lower degree than D, we still sample D neighbors.)
2. For each $i \in [D]$, sample an estimate $y_i \sim Y_{u_i}$. (If $u_i = u_j$, y_i and y_j are still sampled independently.)
3. Use the sum $\sum_{i=1}^{D} y_i$ together with the degree statistics to compute an estimate of x_v via a similar procedure to the [14, 16] algorithm.

Note that each y_i can deviate from x_{u_i}; our new estimate for x_v combines these y_i's and may deviate further from x_v if the errors compound in the right way. Further, we must track the error probabilities, since each of the y_i's might deviate too much from x_{u_i}, and the sample $u_1, \ldots, u_D$ itself might not be representative.

The heart of our multi-pass algorithm is an analysis which manages these compounding errors. Once this analysis is complete, the resulting robust local algorithm is simple to implement in the streaming setting with $O(k)$ passes: In each pass, we start with a "layer" of vertices, measure their degree statistics, and sample D random neighbors for each, which in turn form the next layer.

1.5 Future directions The most immediate future direction is to try to extend our single-pass bounded-degree algorithms (Theorems 1.1 and 1.2) to the setting of general graphs (i.e., without the bounded-degree assumption). A natural starting point would be the machinery we develop for the *multi*-pass algorithm (Theorem 1.3) which eliminates the bounded-degree assumption there. However, there appear to be significant technical challenges, fundamentally because the number of underlying isomorphism classes of vertices' neighborhoods no longer has constant size. We essentially get around this in the multi-pass case by "randomly truncating" the neighborhoods of high-degree vertices, so that we recurse only a random, constant-sized sample of their full neighborhoods. It is not clear how to combine this technique with the mechanisms we develop in the single-pass setting.

Another interesting question is whether the algorithms developed in this paper could be adapted into *quantum* streaming algorithms, just as recent work of [33] adapted the algorithm of [49]. Finally, it would be very interesting to design local algorithms for other CSPs aside from Max-DICUT (such as Max-k-AND) and to generalize our streaming results to these problems.

This version This proceedings version of the paper omits the sections on the random-ordering algorithm (Theorem 1.2) and the multi-pass algorithm (Theorem 1.3). See the forthcoming full version on arXiv for these details.

Outline of this version In § 2, we write some notations for and basic facts about multisets, graphs, probability, and total variation distance which we employ in the paper. In § 3, we develop a notion of the "neighborhood type" of an edge and state a key connection (Theorem 3.1) between the neighborhood type of a uniformly random edge in a graph and approximations of the Max-DICUT value of the same graph. We use this connection in § 4 to design an algorithm for the single-pass unbounded-degree adversarial-ordering settings, thereby proving Theorem 1.1. This algorithm is based on implementing an estimator for the "edge neighborhood-type distribution" defined in the previous section.

2 Preliminaries

For $k \in \mathbb{N}$, $[k]$ denotes the natural numbers between 1 and k inclusive. Given a function $f : S \to T$ and a subset $U \subseteq S$, $f|_U : U \to T$ denotes the restriction of f to U.

For a finite set S, $\mathbf{Dists}(S)$ denotes the set of all probability distributions over S; for $\mathcal{D} \in \mathbf{Dists}(S)$, $\mathcal{D}(S)$ denotes $\Pr_{s' \sim \mathcal{D}}[s' = s]$.

2.1 Multisets A *multiset* may contain multiple (finitely many) copies of elements; for a multiset S, the *multiplicity* of s, denoted $\mathsf{mult}_S(s)$, is the number of copies of s in S; $\mathsf{set}(S)$ is the conversion of S to a set (by forgetting the multiplicity information); and $|S| = \sum_{s \in \mathsf{set}(S)} \mathsf{mult}_S(s)$ is the total number of elements of S.[6] If S and T are multisets, we write $S \subseteq T$ to denote that with $\mathsf{set}(S) \subseteq \mathsf{set}(T)$ and $\mathsf{mult}_S(s) \leq \mathsf{mult}_T(s)$ for all $s \in \mathsf{set}(S)$. If S is a multiset, then $\mathsf{Unif}(S) \in \mathbf{Dists}(\mathsf{set}(S))$ is the distribution over $\mathsf{set}(S)$ which takes value $s \in \mathsf{set}(S)$ with $\frac{1}{|S|}\mathsf{mult}_S(s)$.

For a multiset S, $\mathbf{Ords}(S)$ denotes the set of *orderings* on S, i.e., functions $\sigma : [|S|] \to \mathsf{set}(S)$ such that $|\{i \in [n] : \sigma(i) = s\}| = \mathsf{mult}_S(s)$ for every $s \in \mathsf{set}(S)$.

2.2 Graphs In this paper, a *graph* is a *directed multi*graph, i.e., $G = (V, E)$ for a finite set of *vertices* V and a *multiset* $E \subset V \times V \setminus \{(v, v) : v \in V\}$ of *edges*. For an edge $e = (u, v)$, we let $\mathsf{ends}(e) := \{u, v\}$ denote the set of e's endpoints.

Let $G = (V, E)$ be a graph. The *in-degree* and *out-degree* of v are

$$\mathsf{indeg}_G(v) := \sum_{u \in V} \mathsf{mult}_G(u, v) \qquad \mathsf{outdeg}_G(v) := \sum_{u \in V} \mathsf{mult}_G(v, u),$$

respectively, and the *total degree* (or just *degree*) of v is

$$\mathsf{deg}_G(v) := \sum_{u \in V} (\mathsf{mult}_G(u, v) + \mathsf{mult}_G(v, u)) = \mathsf{outdeg}_G(v) + \mathsf{indeg}_G(v).$$

[6] A multiset S is formally a pair $(\mathsf{set}(S), \mathsf{mult}_S(\cdot) : \mathsf{set}(S) \to \mathbb{N})$.

The *maximum degree* of G is the maximum degree of any vertex. G is *D-bounded* if its maximum degree is at most D.

Let $G = (V, E)$ be a graph and let $x : V \to \{0, 1\}$ be a labeling of G's vertices by Boolean values. The Max-DICUT *value* of x on G is

$$\mathsf{val}_G(x) := \frac{1}{|E|} \sum_{(u,v)\in E} x(u)(1 - x(v)), \tag{2.1}$$

(the sum is counted with multiplicity). (An edge (u, v) is *satisfied* by x if $x(u)(1 - x(v)) = 1$, i.e., if $x(u) = 1$ and $x(v) = 0$. In this sense, the DICUT value is the fraction of satisfied edges.)

Further, we use Eq. (2.1) to define $\mathsf{val}_G(x)$ for "fractional" assignments $x : V \to [0, 1]$ in the natural way. Observe that $\mathsf{val}_G(x)$ equals the expected value of the Boolean assignment that assigns v to 1 w.p. $x(v)$ and 0 w.p. $1 - x(v)$. Hence by averaging:

PROPOSITION 2.1. *Let $G = (V, E)$ be any graph and $x : V \to [0, 1]$ any fractional assignment. Then there exist Boolean assignments $y, z : V \to \{0, 1\}$ such that*

$$\mathsf{val}_G(y) \le \mathsf{val}_G(x) \le \mathsf{val}_G(z).$$

The Max-DICUT *value* of G, without respect to a specific assignment x, is

$$\mathsf{maxval}_G := \max_{x:V\to\{0,1\}} \mathsf{val}_G(x). \tag{2.2}$$

Let $G = (V, E)$. The *induced subgraph* of G on a subset of vertices $U \subseteq V$ is the graph $G[U] = (U, E[U])$ where $E[U]$ is the subset of edges in E with both endpoints in U (with multiplicity). We'll need the following simple fact about induced subgraphs:

PROPOSITION 2.2. *Let $G = (V, E)$, $U \subseteq V$, and $W \subseteq U$. Then $(G[U])[W] = G[W]$.*

Let $G = (V, E)$ and $u, v \in V$. An *path* from u to v is a sequence of vertices $u = w_0, w_1, \ldots, w_{\ell-1}, w_\ell = v \in V$ such that for each $i \in [\ell]$, $(w_{i-1}, w_i) \in E$ or $(w_i, w_{i-1}) \in E$.[7] For $v \in V$, we let the "radius-ℓ ball" around v be

$$\mathsf{ball}^\ell_G(v) := \{u \in V : \exists \text{ a path of length} \le \ell \text{ between } u \text{ and } v \text{ in } G\}.$$

We use the following loose bound on the size of these balls:

PROPOSITION 2.3. *For every $\ell, D \in \mathbb{N}$, $D \ge 2$, if $G = (V, E)$ has maximum degree D, then for every $v \in V$, $|\mathsf{ball}^\ell_G(v)| \le 2D^\ell$.*

Proof. The number of paths originating at v is at most $1 + D + \cdots + D^\ell$. Using the geometric sum formula, this quantity equals $\frac{D^{\ell+1}-1}{D-1} \le \frac{D^{\ell+1}}{D-1} \le 2D^\ell$ (since $D - 1 \ge D/2$). ∎

A *(proper) k-coloring* of $G = (V, E)$ is a function $\chi : V \to [k]$ such that for all $e = (u, v) \in E$, $\chi(u) \ne \chi(v)$. (A coloring is any general function $\chi : V \to [k]$.)

DEFINITION 2.1. (COLORED GRAPH) *A* (properly) k-colored graph *is a pair $(G = (V, E), \chi : V \to [k])$, where G is a graph and χ is a (proper) k-coloring of G.*

2.3 Probability For two probability distributions $\mathcal{X}, \mathcal{Y} \in \mathbf{Dists}(S)$, the *total variation distance* between the distributions is

$$\mathsf{tvdist}(\mathcal{X}, \mathcal{Y}) := \frac{1}{2} \sum_{s\in S} |\mathcal{X}(s) - \mathcal{Y}(s)|. \tag{2.3}$$

We use some standard facts about the distance:

[7] Note that in this notion of "paths", the directions of edges are ignored.

PROPOSITION 2.4. *Let $\mathcal{X}, \mathcal{Y}$ be two probability distributions with $\mathsf{tvdist}(\mathcal{X}, \mathcal{Y}) \le \epsilon$ supported on a finite set S and let $f : S \to [0,1]$ be any function. Then*

$$\left| \mathop{\mathbb{E}}_{s\sim\mathcal{X}}[f(s)] - \mathop{\mathbb{E}}_{s\sim\mathcal{Y}}[f(s)] \right| \le \epsilon.$$

Given a set $s_1, \dots, s_t \in S$, we define the *empirical distribution* $\mathsf{EmpDist}_S(s_1, \dots, s_t) \in \mathbf{Dists}(S)$ via $\mathsf{EmpDist}_S(s_1, \dots, s_t)(s) := \frac{1}{t}|\{i \in [t] : s_i = s\}|$.

PROPOSITION 2.5. *For every finite set S and $\epsilon, \delta > 0$, there exists $t \in \mathbb{N}$ such that the following holds. Let $\mathcal{D}$ be any distribution over S. Then w.p. $1-\delta$ over t independent samples $s_1, \dots, s_t \sim \mathcal{D}$, $\mathsf{tvdist}(\mathcal{D}, \mathsf{EmpDist}_S(s_1, \dots, s_t)) \le \epsilon$.*

Also, for finite sets S, T, a function $\mathcal{F} : S \to \mathbf{Dists}(T)$, and a distribution $\mathcal{D} \in \mathbf{Dists}(S)$, $\mathcal{F} \circ \mathcal{D}$ denotes the "composite" random variable which samples $s \sim \mathcal{D}$ then outputs a sample from $\mathcal{F}(s)$.

PROPOSITION 2.6. ("DATA PROCESSING INEQUALITY") *Let S, T be finite sets and $\mathcal{F} : S \to \mathbf{Dists}(T)$ any function. Further, let $\mathcal{D}_1, \mathcal{D}_2 \in \mathbf{Dists}(S)$. Then*

$$\mathsf{tvdist}(\mathcal{F} \circ \mathcal{D}_1, \mathcal{F} \circ \mathcal{D}_2) \le \mathsf{tvdist}(\mathcal{D}_1, \mathcal{D}_2).$$

PROPOSITION 2.7. *Let S, T be finite sets and $\mathcal{F}, \mathcal{G} : S \to \mathbf{Dists}(T)$ any functions. Let $\mathcal{D} \in \mathbf{Dists}(S)$. Then*

$$\mathsf{tvdist}(\mathcal{F} \circ \mathcal{D}, \mathcal{G} \circ \mathcal{D}) \le \Pr_{s\sim\mathcal{D}}[\mathcal{F}(s) \ne \mathcal{G}(s)].$$

Also, for $t \le |S|$, we define $\mathsf{NoReplace}_t(S)$ as the distribution over $(\mathsf{set}(S))^t$ which iteratively samples $s_1, \dots, s_t \in \mathsf{set}(S)$ via $s_1 \sim \mathsf{Unif}(S)$ and $s_i \sim \mathsf{Unif}(S \setminus \{s_1, \dots, s_{i-1}\})$.[8] We contrast this with the standard "with replacement" product distribution $(\mathsf{Unif}(S))^t$. We have some more standard facts:

PROPOSITION 2.8. ("WITH REPLACEMENT" VS. "WITHOUT REPLACEMENT" SAMPLING) *For every $t \in \mathbb{N}$ and $\epsilon > 0$, there exists $m \in \mathbb{N}$ such that for every multiset S with $|S| \ge m$,*

$$\mathsf{tvdist}(\mathsf{NoReplace}_t(S), (\mathsf{Unif}(S))^t) \le \epsilon.$$

Note that Proposition 2.8 is stated for multisets. However, the statement for multisets reduces immediately to the statement for sets: Given a multiset S with $|S| = n$, consider an arbitrary ordering $\sigma \in \mathbf{Ords}(S)$. Then observe that $\sigma^t \circ \mathsf{NoReplace}_t([n]) = \mathsf{NoReplace}_t(S)$ and $\sigma^t \circ (\mathsf{Unif}([n]))^t = (\mathsf{Unif}(S))^t$ (where $\sigma^t \circ \mathcal{D}$ means to sample $(i_1, \dots, i_t) \sim \mathcal{D}$ and then output $(\sigma(i_1), \dots, \sigma(i_t))$), and apply the data processing inequality.

PROPOSITION 2.9. (HOEFFDING'S INEQUALITY) *Let $X_1, \dots, X_n$ be independent random variables such that $a_i \le X_i \le b_i$ for all $i \in [n]$. For all $t > 0$, we have*

$$\Pr\left(\left| \sum_{i=1}^n X_i - \sum_{i=1}^n \mathbb{E}[X_i] \right| \ge t \right) \le 2 \cdot \mathrm{e}^{-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}}.$$

PROPOSITION 2.10. (FIXING NORMALIZATION) *Let S be a finite set and $\mathcal{D} \in \mathbf{Dists}(S)$ a distribution. Let $Y : S \to \mathbb{R}_{\ge 0}$ be a function such that*

$$\sum_{s\in S} |Y(s) - \mathcal{D}(s)| \le \epsilon.$$

Let $\gamma = \sum_{s\in S} Y(s)$. Then $\widehat{\mathcal{D}}(s) = \frac{Y(s)}{\gamma}$ is a distribution with $\mathsf{tvdist}(\mathcal{D}, \widehat{\mathcal{D}}) \le \epsilon$.

Proof. First, by the triangle inequality $|\gamma - 1| = \left|\sum_{s\in S} Y(s) - 1\right| = \left|\sum_{s\in S}(Y(s) - \mathcal{D}(s))\right| \le \sum_{s\in S}|Y(s) - \mathcal{D}(s)| \le \epsilon$. Further, $\sum_{s\in S}\left|\widehat{\mathcal{D}}(s) - Y(s)\right| = \sum_{s\in S}\left|\frac{1}{\gamma}Y(s) - Y(s)\right| = \sum_{s\in S} Y(s)\left|\frac{1}{\gamma} - 1\right| = \gamma\left|\frac{1}{\gamma} - 1\right| = |1 - \gamma| \le \epsilon$. Hence finally, again using the triangle inequality: $\sum_{s\in S}\left|\widehat{\mathcal{D}}(s) - \mathcal{D}(s)\right| \le \sum_{s\in S}\left(\left|\widehat{\mathcal{D}}(s) - Y(s)\right| + |Y(s) - \mathcal{D}(s)|\right) \le 2\epsilon$. ∎

[8] For multisets S and T, we write $S \supseteq T$ iff for every $x \in T$, $\mathsf{mult}_S(x) \ge \mathsf{mult}_T(x)$. In this case, $S \setminus T$ denotes the new multiset where for every $x \in S$, $\mathsf{mult}_{S\setminus T}(x) = \begin{cases} \mathsf{mult}_S(x) - \mathsf{mult}_T(x) & x \in T \\ \mathsf{mult}_S(x) & x \notin T \end{cases}$. Thus, in particular, $|S \setminus T| = |S| - |T|$ where $|\cdot|$ denotes cardinality, i.e., the sum of multiplicities.

3 Algorithms from neighborhood type sampling

In this section, we develop some general techniques for reducing $(1/2-\epsilon)$-approximating the Max-DICUT value of graphs to estimating certain neighborhood-type distributions in graphs.

3.1 Preprocessing colors

PROPOSITION 3.1. *Let $G=(V,E)$ be a graph and $k \geq 2 \in \mathbb{N}$, and let $\delta > 0$. Let $\mathcal{K}$ be any two-wise independent distribution over functions $\chi : V \to [k]$. (I.e., for all $u \neq v \in V$, and $i,j \in [k]$, $\Pr_{\chi\sim\mathcal{K}}[\chi(u)=i \wedge \chi(v)=j] = 1/k^2$.) Then,*

$$\Pr_{\chi\sim\mathcal{K}}\left[\Pr_{e=(u,v)\sim\mathsf{Unif}(E)}[\chi(u)=\chi(v)] \geq \delta\right] \leq \frac{1}{\delta k}.$$

In particular, the RHS is less than 1% if $k \geq 100/\delta$.

Proof. Enumerate E's edges as $\{e_1,\ldots,e_m\}$, and let X_j be the indicator for the event that $\chi(u_j)=\chi(v_j)$ (where $e_j=(u_j,v_j)$). Then by 2-wise independence, $\Pr_\chi[X_j] = \sum_{c=1}^{k} \Pr_\chi[\chi(u_j)=\chi(v_j)=c] = k \cdot 1/k^2 = 1/k$. Then $\Pr_{e=(u,v)\sim\mathsf{Unif}(E)}[\chi(u)=\chi(v)] = \frac{1}{m}\sum_{j=1}^{m} X_j$, and therefore $\mathbb{E}_\chi[\Pr_{e=(u,v)\sim\mathsf{Unif}(E)}[\chi(u)=\chi(v)]] = 1/k$ by linearity of expectation. Finally, we apply Markov's inequality. ∎

We use this proposition to add a "preprocessing" step to all of our algorithms: Before we start running a streaming algorithm $\mathcal{A}$, we sample a 2-wise independent function $\chi : V \to [k]$; we give $\mathcal{A}$ black-box access to χ (encoded as a polynomial's coefficients, which requires polylogarithmic bits); then immediately before each edge (u,v) is processed by $\mathcal{A}$, we discard the edge (i.e., do not pass it to $\mathcal{A}$) if $\chi(u)=\chi(v)$. The input from $\mathcal{A}$'s point of view is then a stream of graph edges together with black-box access to a proper coloring. Moreover, in the case of random-ordering algorithms, note that conditioned on χ, we still provide a uniformly random ordering over the graph's remaining edges. In other words, the distributions "sample a uniform random ordering of all edges, then sample a coloring and throw away the improperly-colored edges and output the remaining edges in order" and "sample a coloring and throw away the improperly-colored edges, then sample a uniformly random ordering of the remaining edges" are identical.

3.2 Induced subgraphs and neighborhoods We will require the following useful proposition about neighborhoods inside of induced subgraphs:

PROPOSITION 3.2. *Let $k,\ell \in \mathbb{N}$ and let $(G=(V,E),\chi : V \to [k])$ be a k-colored graph. Let $S \subseteq V$ with $v \in S$. The following are equivalent:*

1. *Every vertex $w \in \mathsf{ball}^{\ell-1}_{G[S]}(v)$ has $\mathsf{deg}_{G[S]}(w) = \mathsf{deg}_G(w)$.*
2. *$\mathsf{ball}^{\ell}_{G}(v) \subseteq S$.*
3. *$\mathsf{ball}^{\ell}_{G[S]}(v) = \mathsf{ball}^{\ell}_{G}(v)$.*

Proof. Observe that by the definition of induced subgraph, for every $w \in S$, $\mathsf{deg}_{G[S]}(w) \leq \mathsf{deg}_G(w)$. Further, $\mathsf{ball}^{\ell}_{G[S]}(v) \subseteq \mathsf{ball}^{\ell}_{G}(v)$.

($2 \implies 1$) Suppose there exists a vertex $w \in \mathsf{ball}^{\ell-1}_{G[S]}(v)$ with $\mathsf{deg}_{G[S]}(w) < \mathsf{deg}_G(w)$. Then there exists some vertex $z \in V$ incident to w but not in S. But $w \in \mathsf{ball}^{\ell-1}_{G}(v)$ so $z \in \mathsf{ball}^{\ell}_{G}(v)$ (i.e., w is within distance $\ell-1$ of v so z must be within distance ℓ of v).

($3 \implies 2$) By definition, $\mathsf{ball}^{\ell}_{G[S]}(v) \subseteq S$, hence by assumption $\mathsf{ball}^{\ell}_{G}(v) \subseteq S$.

($1 \implies 3$) Suppose there exists a vertex $z \in \mathsf{ball}^{\ell}_{G}(v) \setminus \mathsf{ball}^{\ell}_{G[S]}(v)$. Thus, there exists some path $v=w_0,\ldots,z=w_L$ in G between v and z of length $L \leq \ell$ (each w_i is incident to w_{i-1}). Note that $v \in \mathsf{ball}^{\ell}_{G[S]}(v)$ while $z \notin \mathsf{ball}^{\ell}_{G[S]}(v)$. Thus, there exists some $i \in [L]$ such that $w_0,\ldots,w_{i-1} \in \mathsf{ball}^{\ell}_{G[S]}(v)$ but $w_i \notin \mathsf{ball}^{\ell}_{G[S]}(v)$. By the former, $w_{i-1} \in \mathsf{ball}^{i-1}_{G[S]}(v)$, so we must have $w_i \notin S$, else we would have $w_i \in \mathsf{ball}^{i}_{G[S]}(v)$, contradicting the latter. Since $w_{i-1} \in S$ but $w_i \notin S$, we deduce that $\mathsf{deg}_{G[S]}(w_{i-1}) < \mathsf{deg}_G(w_{i-1})$. ∎

3.3 Edge-type distributions and the multi-pass algorithm Recall that the local 1/2-approximation for Max-DICUT of [14, 16] builds a random assignment to a k-colored graph $(G = (V, E), \chi : V \to [k])$'s vertices, where each vertex v is assigned independently with a probability depending only on its local neighborhood. To design streaming $(1/2 - \epsilon)$-approximation algorithms for Max-DICUT, we will be interested in using a streaming algorithm to simulate the [14, 16] algorithm, or more precisely, to estimate the value of the cut it produces. This is equivalent to estimating the probability that a random edge in the graph is satisfied by the cut. But the probability that the algorithm's assignment satisfies a particular edge depends only on the neighborhood-types of its two endpoints. So we arrive at the goal of estimating the distribution of the endpoints' $(k+1)$-neighborhood types of a random edge.

Definition 3.1. (Doubly-rooted colored graph) *A* doubly rooted (properly) k-colored graph *is a triple* $(G = (V, E), \chi : V \to [k], e \in E)$*, where* G *is a graph,* χ *is a proper* k*-coloring of* G*, and* e *is a designated* root *edge.*

For a bijection $\phi : V \to V'$ and an edge $e = (u, v)$, we use $\phi(e)$ to denote the pair $(\phi(u), \phi(v))$.

Definition 3.2. (Isomorphism of doubly rooted colored graphs) *Two doubly rooted* k*-colored graphs* $(G = (V, E), \chi, e)$ *and* $(G' = (V', E'), \chi', e')$ *are* isomorphic *if there exists a bijection* $\phi : V \to V'$ *such that (i) for all* $u \neq v \in V$, $\mathsf{mult}_E(u, v) = \mathsf{mult}_{E'}(\phi(u, v))$*, (ii) for all* $v \in V$, $\chi(v) = \chi'(\phi(v))$*, and (iii)* $\phi(e) = e'$*.*

Note that, importantly, this notion of isomorphism does *not* allow exchanging colors.[9] Also, isomorphism preserves the degrees of vertices, the distances between pairs of vertices, and the Max-DICUT values of assignments.

Given a doubly rooted k-colored graph (G, χ, e), let $\mathsf{type}(G, \chi, e)$ denote the corresponding isomorphism class ("type") of doubly rooted k-colored graphs. For $k, \ell, D \in \mathbb{N}$, we let $\mathbf{Typ}_k^{\ell,D}$ denote the set of all isomorphism classes of D-bounded doubly rooted k-colored graphs where every vertex is distance $\leq \ell$ from (at least) one of the roots. $\mathbf{Typ}_k^{\ell,D}$ is a finite set by Proposition 2.3; we let $\mathbf{N}_k^{\ell,D} := |\mathbf{Typ}_k^{\ell,D}|$ denote its size.

Given $G = (V, E)$ and an edge $e = (u, v) \in E$, we let $\mathsf{ball}_G^\ell(e) := \mathsf{ball}_G^\ell(u) \cup \mathsf{ball}_G^\ell(v)$.

Definition 3.3. (radius-ℓ neighborhood type of edge) *Let* $k, \ell \in \mathbb{N}$, $G = (V, E)$, $\chi : V \to [k]$*, and* $e \in E$*. The* radius-ℓ neighborhood type *of* e*, denoted* $\mathsf{nbhdtype}_{G,\chi}^\ell(e)$*, is* $\mathsf{nbhdtype}_{G,\chi}^\ell(e) := \mathsf{type}(G[N], \chi|_N, e)$ *where* $N := \mathsf{ball}_G^\ell(e)$.

If G is D-bounded then $\mathsf{nbhdtype}_{G,\chi}^\ell(e) \in \mathbf{Typ}_k^{\ell,D}$.

We also require a variant of Proposition 3.2 for doubly-rooted graphs:

Proposition 3.3. *Let* $k, \ell \in \mathbb{N}$ *and let* $(G = (V, E), \chi : V \to [k])$ *be a* k*-colored graph. Let* $S \subseteq V$ *with* $e = (u, v) \in E$ *and* $u, v \in S$*. The following are equivalent:*

1. *Every vertex* $w \in \mathsf{ball}_{G[S]}^{\ell-1}(e)$ *has* $\mathsf{deg}_{G[S]}(w) = \mathsf{deg}_G(w)$.
2. $\mathsf{ball}_G^\ell(e) \subseteq S$.
3. $\mathsf{ball}_{G[S]}^\ell(u) = \mathsf{ball}_G^\ell(v)$ *and* $\mathsf{ball}_{G[S]}^\ell(v) = \mathsf{ball}_G^\ell(v)$.

Further, if any of these equivalent conditions holds, then $\mathsf{nbhdtype}_{G[S],\chi|_S}^\ell(u, v) = \mathsf{nbhdtype}_{G,\chi}^\ell(u, v)$.

Proof. The equivalence of the first three conditions follows immediately from Proposition 3.2: Each condition is the conjunction of the two corresponding conditions from Proposition 3.2 for u and v (e.g., $\mathsf{ball}_G^\ell(u, v) \subseteq S$ iff $\mathsf{ball}_G^\ell(u) \subseteq S$ and $\mathsf{ball}_G^\ell(v) \subseteq S$). For the final implication, we reproduce the proof from Proposition 3.2: By definition, $\mathsf{nbhdtype}_{G[S],\chi|_S}^\ell(u, v) = \mathsf{type}((G[S])[N], \chi|_S|_N, u, v)$ where $N = \mathsf{ball}_{G[S]}^\ell(u, v)$, while $\mathsf{nbhdtype}_{G,\chi}^\ell(u, v) = \mathsf{type}(G[M], \chi|_M, u, v)$ where $M = \mathsf{ball}_G^\ell(u, v)$. By assumption, $N = M$, so $G[M] = G[N] = (G[S])[N]$ by Proposition 2.2 (and trivially $\chi|_S|_N = \chi|_N$). ∎

[9] For instance, an isolated vertex colored 1 is *not* isomorphic to an isolated vertex colored 2.

PROPOSITION 3.4. *For all $D, \ell \in \mathbb{N}$, there exists $\Delta \in \mathbb{N}$ with the following property. For every graph $G = (V, E)$ with maximum-degree D and every $e \in E$,*

$$|\{e' \in E : \mathsf{ball}^{\ell}_G(e) \cap \mathsf{ball}^{\ell}_G(e') \neq \emptyset\}| \leq \Delta.$$

Proof. If $\mathsf{ball}^{\ell}_G(e) \cap \mathsf{ball}^{\ell}_G(e') \neq \emptyset$, there must be $w \in \mathsf{ends}(e)$ and $w' \in \mathsf{ends}(e')$ such that $w' \in \mathsf{ball}^{2\ell}_G(w)$; hence, $v' \in \mathsf{ball}^{2\ell+2}_G(v)$ where v, v' are arbitrary vertices in $\mathsf{ends}(e)$ and $\mathsf{ends}(e')$, respectively. By Proposition 2.3, $|\mathsf{ball}^{2\ell+2}_G(v)| \leq 2D^{2\ell+3}$. Further, for any such v', there can be at most D neighbors in E (by the maximum-degree assumption). This gives the bound for $\Delta := 2D^{2\ell+4}$. ∎

3.4 Edge-type distribution Now, we define a notion of the *edge-type distribution* in a graph, and describe how estimating this distribution suffices for approximating the Max-DICUT value of a graph.

DEFINITION 3.4. (EDGE-TYPE DISTRIBUTION) *Let $k, \ell, D \in \mathbb{N}$ and $(G = (V, E, \chi : V \to [k])$ be a D-bounded k-colored graph. The* radius-ℓ neighborhood type distribution *of (G, χ), denoted $\mathsf{EdgeNbhdTypeDist}^{\ell}_{G;\chi}$, is the distribution over $\mathbf{Typ}^{\ell,D}_k$ given by sampling a random $e \sim \mathsf{Unif}(E)$ and outputting $\mathsf{nbhdtype}^{\ell}_{G,\chi}(e)$.*

The fact that the edge-type distribution suffices for approximating the Max-DICUT value of a graph is captured by the following theorem:

THEOREM 3.1. (IMPLIED BY [14, 16]) *Let $k, D \in \mathbb{N}$. There exists a function $\mathtt{Local} : \mathbf{Typ}^{k,D}_k \to [0, 1]$ such that the following holds. Let $(G = (V, E), \chi : V \to [k])$ be a D-bounded k-colored graph. Then*

$$\frac{1}{2}\mathsf{maxval}_G \leq \mathop{\mathbb{E}}_{T \sim \mathsf{EdgeNbhdTypeDist}^{k}_{G;\chi}} [\mathtt{Local}(T)] \leq \mathsf{maxval}_G.$$

We remark that this theorem is not stated explicitly in the papers [14, 16], but is directly implied by these works. The key fact is that in the deterministic algorithm for producing a fractional cut presented in [14, §4], the assignment to each vertex depends only on the isomorphism class of the radius-k neighborhood of that vertex; thus, the probability any edge is satisfied depends only on the isomorphism classes of the neighborhoods of its two endpoints, and the type of the edge is only more informative (it contains additional information about the intersection of these two neighborhoods).

Immediately from properties of the total variation distance (in particular, Proposition 2.4), we deduce:

COROLLARY 3.1. *Let $k, D \in \mathbb{N}$ and let $\mathtt{Local} : \mathbf{Typ}^{k,D}_k \to [0, 1]$ be the function in the previous theorem. For every $\epsilon > 0$ and $(G = (V, E), \chi : V \to [k])$ a D-bounded k-colored graph, if $\mathcal{D} \in \mathbf{Dists}(\mathbf{Typ}^{k,D}_k)$ is such that $\mathsf{tvdist}(\mathcal{D}, \mathsf{EdgeNbhdTypeDist}^{k}_{G;\chi}) \leq \epsilon$, then*

$$\frac{1}{2}\mathsf{maxval}_G - 2\epsilon \leq \mathop{\mathbb{E}}_{T \sim \mathcal{D}} [\mathtt{Local}(T)] - \epsilon \leq \mathsf{maxval}_G.$$

Now, we arrive at the main statements on estimating the edge-type distribution which we develop in the following sections.

THEOREM 3.2. (SINGLE-PASS ADVERSARIAL-ORDER ESTIMATOR) *For all $k, \ell, D \in \mathbb{N}$ and $\epsilon, \delta > 0$, there exists $c > 0$ such that the following holds. There exists an $O(n^{1-c})$-space streaming algorithm that, for every D-bounded k-colored graph $(G = ([n], E), \chi : [n] \to [k])$, given (black-box access to) χ and a single, adversarially-ordered pass over G's edges, outputs $\mathcal{D} \in \mathbf{Dists}(\mathbf{Typ}^{\ell,D}_k)$ satisfying $\mathsf{tvdist}(\mathcal{D}, \mathsf{EdgeNbhdTypeDist}^{\ell}_{G;\chi}) \leq \epsilon$ except w.p. δ.*

THEOREM 3.3. (SINGLE-PASS RANDOM-ORDER ESTIMATOR) *For all $k, \ell, D \in \mathbb{N}$ and $\epsilon, \delta > 0$, there exists $C > 0$ such that the following holds. There exists a $C \log n$-space streaming algorithm that, for every D-bounded k-colored graph $(G = ([n], E), \chi : [n] \to [k])$, given (black-box access to) χ and a single, randomly-ordered pass over G's edges, outputs $\mathcal{D} \in \mathbf{Dists}(\mathbf{Typ}^{\ell,D}_k)$ satisfying $\mathsf{tvdist}(\mathcal{D}, \mathsf{EdgeNbhdTypeDist}^{\ell}_{G;\chi}) \leq \epsilon$ except w.p. δ.*

We are now ready to prove Theorem 1.1 and Theorem 1.2, assuming Theorem 3.2 and Theorem 3.3.

Proof. [Proof of Theorem 1.1] Let $h : [n] \to [100/\epsilon]$ be a 2-wise independent hash function that describes a coloring $\chi : V \to [100/\epsilon]$. It follows from Proposition 3.1 that with probability at least 9/10, the fraction of monochromatic edges is at most $\epsilon/2$. We "delete" these monochromatic edges from the stream and compute the Max-DICUT value of the remaining graph $\widetilde{G}$. With probability at least 9/10, $|\mathsf{maxval}_{\widetilde{G}} - \mathsf{maxval}_G| \leq \epsilon/2$. For the reduced graph $\widetilde{G}$, the hash function h gives black-box access to a proper coloring of its vertices. Thus, we can apply the algorithm from Theorem 3.2 to output a distribution $\mathcal{D} \in \mathbf{Dists}(\mathbf{Typ}_k^{\ell,D})$ satisfying $\mathsf{tvdist}(\mathcal{D}, \mathsf{EdgeNbhdTypeDist}^{\ell}_{\widetilde{G};\chi}) \leq \epsilon/4$ except w.p. 1/10. It follows from Corollary 3.1 that

$$\frac{1}{2}\mathsf{maxval}_{\widetilde{G}} - \epsilon/2 \leq \mathop{\mathbb{E}}_{T\sim\mathcal{D}}[\mathtt{Local}(T)] - \epsilon/4 \leq \mathsf{maxval}_{\widetilde{G}}.$$

Applying the union bound, we conclude that there is a streaming algorithm which $(1/2-\epsilon)$-approximates the Max-DICUT value of a D-bounded n-vertex graph in $O(n^{1-\Omega(1)})$ space using a single, adversarially-ordered pass over the list of edges, with probability at least 8/10. ∎

The proof of Theorem 1.2 is analogous.

4 Adversarial-ordering, $o(n)$-space, single-pass algorithm

In this section, we prove Theorem 3.2, using Algorithm 1 that we describe below. The algorithm uses Algorithm 2 as a sub-routine. In addition to the inputs given to Theorem 3.2, the latter algorithm takes as input a target type $T \in \mathbf{Typ}_k^{\ell,D}$, and an estimate $\widehat{m} \in \mathbb{N}$ for the number of edges in the graph, and outputs an estimate for the probability mass of T in $\mathsf{EdgeNbhdTypeDist}^{\ell}_{G;\chi}$.

Algorithm 1 $\mathsf{Bounded\text{-}Degree\text{-}Adversarial}_D(n, k, \chi, \ell, \epsilon, \delta, \boldsymbol{\sigma})$

Parameters: Number of vertices $n \in \mathbb{N}$, number of colors $k \in \mathbb{N}$, coloring $\chi : [n] \to [k]$, maximum degree $D \in \mathbb{N}$, radius $\ell \in \mathbb{N}$, accuracy $\epsilon > 0$, and failure probability $\delta > 0$.
Input: A stream of edges $\boldsymbol{\sigma}$ from G

Maintain a global counter for the number of edges m
for every integer b from 0 to $\lfloor \log(nD/2) \rfloor$ **do**
  for every type $T \in \mathbf{Typ}_k^{\ell,D}$ **do**
    $Y_{b,T} \leftarrow \mathsf{Bounded\text{-}Degree\text{-}Adversarial\text{-}FixedType}_D(n, k, \chi, \ell, \epsilon/\mathbf{N}_k^{\ell,D}, \delta/\mathbf{N}_k^{\ell,D}, \boldsymbol{\sigma}, T, 2^b)$
  end for
end for
$\gamma \leftarrow \sum_{T\in\mathbf{Typ}_k^{\ell,D}} Y_{b,T}$
$\mathcal{N} \leftarrow (\frac{1}{\gamma} Y_{b,T})_{T\in\mathbf{Typ}_k^{\ell,D}}$ where $b = \lfloor \log m \rfloor$
Output $\mathcal{N}$

Given a type $T \in \mathbf{Typ}_k^{\ell,D}$, and an arbitrary representative (G, χ, r, s) of T, let $a(T) := |\mathsf{ball}_G^{\ell}(r, s)|$ (one easily verifies that this does not depend on the choice of representative).

Algorithm 2 Bounded-Degree-Adversarial-FixedType$_D(n, k, \chi, \ell, \epsilon, \delta, \boldsymbol{\sigma}, T, \widehat{m})$

Parameters: Number of vertices $n \in \mathbb{N}$, number of colors $k \in \mathbb{N}$, coloring $\chi : [n] \to [k]$, maximum degree $D \in \mathbb{N}$, radius $\ell \in \mathbb{N}$, accuracy $\epsilon > 0$, failure probability $\delta > 0$, target type $T \in \mathbf{Typ}_k^{\ell,D}$, and estimate $\widehat{m} \in \mathbb{N}$ for number of edges.
Input: A stream of edges $\boldsymbol{\sigma}$ from G.

Pre-processing:
Set Δ to be the constant given by Proposition 3.4.
Set $K \leftarrow ((\epsilon^2 \delta \widehat{m})/(2D\Delta))^{1/a(T)}$.
if $K \leq 1$ **then**
    Store the entire input stream $\boldsymbol{\sigma}$ and return the exact fraction of type-T edges
end if
Let $H : [n] \to [2^{\lfloor \log_2 K \rfloor}]$ be a $2a(T)$-wise independent hash function
Initialize $S \leftarrow \emptyset$ (set) and $F \leftarrow \emptyset$ (multiset)
Initialize $c \leftarrow 1/(2 \cdot a(T))$

Stream processing:
for edge e in stream **do**
    for $v \in \mathsf{ends}(e)$ **do**
        if $H(v) = 1$ and $v \notin S$ **then**
            $S \leftarrow S \cup \{v\}$
            $\mathtt{degs}[v] \leftarrow 0$
        end if
        if $v \in S$ **then**
            $\mathtt{degs}[v] \leftarrow \mathtt{degs}[v] + 1$
        end if
    end for
    if $\mathsf{ends}(e) \subseteq S$ **then**
        $F \leftarrow F \cup \{e\}$
    end if
    if $|S| \geq n^{1-c}$ **then**
        Terminate and output `fail`
    end if
end for

Post-processing:
$X \leftarrow 0$
for $u \neq v \in V$ **do**
    if $(u, v) \in F$ **then**
        if every vertex $w \in \mathsf{ball}^{\ell-1}_{(S,F)}(u, v)$ has $\mathtt{degs}[w] = \mathsf{deg}_{(S,F)}(w)$ **then**
            if $\mathsf{nbhdtype}^{\ell}_{(S,F),\chi}(u, v) = T$ **then**
                $X \leftarrow X + \mathsf{mult}_F((u, v))$
            end if
        end if
    end if
end for
return $K^{a(T)} X/m$.

The key correctness lemma of this section is the following:

LEMMA 4.1. (CORRECTNESS AND SPACE BOUND FOR ALGORITHM 2) *Let $k, \ell, D \in \mathbb{N}$ and $\epsilon, \delta > 0$ be constants. Let $n \in \mathbb{N}$ and let $(G = (V, E), \chi : V \to [k])$ be a k-colored, D-bounded graph. Let $m = |E|$ and $\widehat{m} \in \mathbb{N}$. For*

every possible ordering $\boldsymbol{\sigma}$ of E, Algorithm 2 runs in $\widetilde{O}(n^{1-c})$ space. Further, if $m/2 < \widehat{m} \le m$, w.p. $\ge 1-\delta$ the algorithm gives an output Y satisfying

$$\left| Y - \mathsf{EdgeNbhdTypeDist}^{\chi}_{G;\ell}(T) \right| \le \epsilon.$$

Proof. It follows from the condition on Line 31 that the Algorithm 2 always uses at most $\widetilde{O}(n^{1-c})$ space (since the maximum degree is at most D, $|F| \le D|S|$).

Now, we analyze the correctness of a *hypothetical* version of Algorithm 2 that does not perform the check in Line 31. We show that this hypothetical version satisfies both the correctness condition and $|S| \le n^{1-c}$ w.p. δ, which in turn implies the correctness result for the real algorithm.

Note that $K > 1$ as long as $\widehat{m}$ is at least a constant; further, when $K \le 1$, the algorithm is guaranteed to be correct deterministically and uses only constant space. We assume hereafter that $K > 1$.

Every non-isolated vertex passes the check in Line 20 with probability $p := 1/2^{\lfloor \log_2 K \rfloor}$. Thus, $2/K \ge p \ge 1/K$.

The effect of the loop on Line 18 is to guarantee the following at the end of the stream:

1. $S = \{v \in [n] : \mathsf{deg}_G(v) > 0 \text{ and } H(v) = 1\}$.
2. $F = E[S]$, i.e., F is the subset of edges in E with both endpoints in S (with multiplicity). Thus, in particular, the pair (S, F) is the induced subgraph $G[S]$.
3. For every $v \in S$, $\mathsf{degs}[v] = \mathsf{deg}_G(v)$.

Thus, Line 38 checks whether every vertex $w \in \mathsf{ball}^{\ell-1}_{G[S]}(u,v)$ has $\mathsf{deg}_{G[S]}(w) = \mathsf{deg}_G(w)$. By Proposition 3.3, this is equivalent to $\mathsf{ball}^{\ell}_{G[S]}(u,v) \subseteq S$. Further, Line 39 checks if $\mathsf{nbhdtype}^{\ell}_{G[S],\chi|_S}(u,v) = T$.

Now, we analyze the random variable X, which controls the output of the algorithm. X is set to 0 on Line 35 and increased on Line 40 for pairs $(u,v), u \ne v$ satisfying certain conditions. Let $X_{(u,v)}$ denote the amount by which X is increased on Line 40 during the (u,v)-iteration, so that $X = \sum_{u \ne v \in V} X_{(u,v)}$. We claim:

Claim 4.1. *For every $u \ne v \in V$, the following holds: If $(u,v) \in E$, $\mathsf{ball}^{\ell}_G(u,v) \subseteq S$, and $\mathsf{nbhdtype}^{\ell}_{G,\chi}(u,v) = T$, then $X_{(u,v)} = \mathsf{mult}_E((u,v))$, and otherwise, $X_{(u,v)} = 0$.*

Proof. Note that $X_{(u,v)} = \mathsf{mult}_F((u,v))$ if and only if the checks on Lines 36, 38 and 39 all pass, and otherwise $X_{(u,v)} = 0$. To begin, since $F \subseteq E$, if $(u,v) \notin E$, then $(u,v) \notin F$, so the check on Line 36 fails and $X_{(u,v)} = 0$. Otherwise, $(u,v) \in E$, and by definition of the induced subgraph, $\mathsf{mult}_E((u,v)) = \mathsf{mult}_F((u,v))$. Next, Line 38 checks whether $\mathsf{ball}^{\ell}_{G[S]}(u,v) \subseteq S$. Finally, Line 39 checks if $\mathsf{nbhdtype}^{\ell}_{G[S],\chi|_S}(u,v) = T$. By Proposition 3.3, $\mathsf{ball}^{\ell}_{G[S]}(u,v) \subseteq S$ implies that $\mathsf{nbhdtype}^{\ell}_{G[S],\chi|_S}(u,v) = \mathsf{nbhdtype}^{\ell}_{G,\chi|_S}(u,v)$, so the check on Line 39 passes if and only if $\mathsf{nbhdtype}^{\ell}_{G,\chi|_S}(u,v) = T$, as desired. ∎

Now let $\mathcal{T} := \{e \in \mathsf{set}(E) : \mathsf{nbhdtype}^{\ell}_{G,\chi}(e) = T\}$ denote the set of edges of type T. (Hence $|\mathcal{T}| = \mathsf{EdgeNbhdTypeDist}^{\chi}_{G;\ell}(T) \cdot m$.) By the claim, $X_{(u,v)}$ is zero unless $(u,v) \in \mathcal{T}$, in which case it equals $\mathsf{mult}_E(e)$ iff $\mathsf{ball}^{\ell}_G(u,v) \subseteq S$. Hence we forget X_e for $e \notin \mathcal{T}$ and write $X = \sum_{e \in \mathcal{T}} X_e$. Further, recall that S is a set which (a) contains each vertex with probability p and (b) is $2a(T)$-wise independent (and in particular $a(T) = |\mathsf{ball}^{\ell}_G(u,v)|$). Thus $\mathbb{E}[X] = p^{a(T)}|\mathcal{T}|$ and $(\mathbb{E}[X])^2 = p^{2a(T)}|\mathcal{T}|^2$. Next, we compute

$$\begin{aligned}
\mathbb{E}[X^2] &= \sum_{e,e' \in \mathcal{T}} \mathbb{E}[X_e X_{e'}] \\
&= \sum_{e,e' \in \mathcal{T} : X_e, X_{e'} \text{ independent}} \mathbb{E}[X_e]\,\mathbb{E}[X_{e'}] + \sum_{e,e' \in \mathcal{T} : X_e, X_{e'} \text{ dependent}} \mathbb{E}[X_e X_{e'}] \\
&\le (\mathbb{E}[X])^2 + \sum_{e,e' \in \mathcal{T} : X_e, X_{e'} \text{ dependent}} \mathbb{E}[X_e X_{e'}]. \qquad (X_e, X_{e'} \text{ non-negative})
\end{aligned}$$

Hence,

$$\mathrm{Var}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X]^2) \le \sum_{e,e' \in \mathcal{T} : X_e, X_{e'} \text{ dependent}} \mathbb{E}[X_e X_{e'}].$$

Now when are X_e and $X_{e'}$ dependent? Suppose $e = (u, v)$ and $e' = (u', v')$. Recall, X_e is determined by whether $\mathsf{ball}^\ell_G(u, v) \subseteq S$ and similarly $X_{e'}$ by whether $\mathsf{ball}^\ell_G(u', v') \subseteq S$. Recall that $|\mathsf{ball}^\ell_G(u, v)| = |\mathsf{ball}^\ell_G(u', v')| = a(T)$ (since both e and e' are assumed to have type T) and therefore, by $2a(T)$-wise independence of S, X_e and $X_{e'}$ are independent unless $\mathsf{ball}^\ell_G(u, v) \cap \mathsf{ball}^\ell_G(u', v') \neq \emptyset$. Hence by Proposition 3.4, X_e is dependent on $X_{e'}$ for at most $2D^{2\ell+4}$ distinct edges e'. Since $X_{e'} \leq D$, we deduce

$$\mathrm{Var}[X] \leq D \sum_{e,e' \in \mathcal{T} : X_e, X_{e'} \text{ dependent}} \mathbb{E}[X_e] \leq 2D^{2\ell+5} \sum_{e \in \mathcal{T}} \mathbb{E}[X_e] = \Delta D \, \mathbb{E}[X] \,,$$

where Δ is the constant from Proposition 3.4.

Set $t := \frac{\epsilon\sqrt{p^{a(T)}m}}{\sqrt{\Delta D}}$, so that

$$t\sqrt{\mathrm{Var}[X]} \leq \frac{\epsilon\sqrt{p^{a(T)}m}}{\sqrt{\Delta D}} \cdot \sqrt{\Delta D \, \mathbb{E}[X]} = \epsilon p^{a(T)} \sqrt{m|\mathcal{T}|} \leq \epsilon p^{a(T)} m.$$

Hence by Chebyshev's inequality,

$$\Pr[|X - \mathbb{E}[X]| \geq \epsilon p^{a(T)} m] \leq \frac{1}{t^2}.$$

Now we expand

$$\frac{1}{t^2} = \frac{\Delta D}{\epsilon^2 p^{a(T)} m} = \frac{\Delta D}{\epsilon^2 (2\Delta D/(\epsilon^2 \delta \widehat{m}) m} = \frac{\delta \widehat{m}}{2m} \leq \delta/2$$

since $\widehat{m} \leq m$.

Finally, let us analyze the number of vertices in S. Note that $\mathbb{E}[|S|] = pn_+$, where n_+ denotes the number of non-isolated vertices in G. Since $p = O(1/m^{1/a(T)})$ by assumption, $pn_+ = O(n_+/m^{1/a(T)}) \leq O(m^{1-1/a(T)}) = O(n^{1-1/a(T)})$ using $n_+ \leq 2m$ and $m \leq Dn$. By Markov's inequality, $\Pr[|S| \geq 2\,\mathbb{E}[|S|]/\delta] \leq \delta/2$. For sufficiently large n, $2\,\mathbb{E}[|S|]/\delta$ is at most n^{1-c}.

Applying the union bound, we conclude that Algorithm 2 succeeds with probability at least $1 - \delta$. ∎

Proof. [Proof of Theorem 3.2] Again, $D, k, \ell, \epsilon, \delta$ are constants independent of n. Since Algorithm 1 makes at most $O(\log n)$ calls to Algorithm 2, by Lemma 4.1, the space usage of Algorithm 1 is at most $\widetilde{O}(n^{1-c})$. We now prove correctness.

By definition, Algorithm 1 outputs $\mathcal{N} \in \mathbf{Dists}(\mathbf{Typ}^{\ell,D}_k)$ where for $T \in \mathbf{Typ}^{\ell,D}_k$,

$$\mathcal{N}(T) = \frac{Y_{b,T}}{\gamma},$$

for $b = \lfloor \log m \rfloor$ where $\gamma = \sum_{T \in \mathbf{Typ}^{\ell,D}_k} Y_{b,T}$. Note that for this value of b, $m/2 < 2^b \leq m$. Since $Y_{b,T}$ is the output of running Algorithm 2 with parameters $(n, k, \chi, \ell, \epsilon/\mathbf{N}^{\ell,D}_k, \delta/\mathbf{N}^{\ell,D}_k, \boldsymbol{\sigma}, T, 2^b)$, by Lemma 4.1, for every $T \in \mathbf{Typ}^{\ell,D}_k$, with failure probability at most $\delta/\mathbf{N}^{\ell,D}_k$, we have

$$|Y_{b,T} - \mathsf{EdgeNbhdTypeDist}^\chi_{G;\ell}(T)| \leq \epsilon/\mathbf{N}^{\ell,D}_k.$$

Thus, by the union bound, with failure probability at most δ,

$$\sum_{T \in \mathbf{Typ}^{\ell,D}_k} |Y_{b,T} - \mathsf{EdgeNbhdTypeDist}^\chi_{G;\ell}(T)| \leq \epsilon.$$

Finally, we apply Proposition 2.10. ∎

Acknowledgments

R.R.S. is supported by the Department of Atomic Energy, Government of India, under project no. RTI4001. N.S. is supported in part by an NSF Graduate Research Fellowship (Award DGE 2140739). M.S. is supported in part by a Simons Investigator Award and NSF Award CCF 2152413. S.V. is supported in part by NSF award

CCF 2348475. Part of the work was conducted when S.V. was visiting the Simons Institute for the Theory of Computing as a research fellow in the Sublinear Algorithms program. Part of the work was conducted when S.V. was a graduate student at Harvard University, and supported in part by a Google Ph.D. Fellowship, a Simons Investigator Award to M.S., and NSF Award CCF 2152413.

We would like to thank Hoaian Nguyen and the anonymous reviewers at SODA for helpful comments on the text.

References

[1] S. Assadi, *Tight Space-Approximation Tradeoff for the Multi-Pass Streaming Set Cover Problem*, in Symposium on Principles of Database Systems (PODS), 2017.

[2] S. Assadi and S. Behnezhad, *Beating two-thirds for random-order streaming matching*, in International Colloquium on Automata, Languages, and Programming (ICALP), 2021.

[3] S. Assadi, Y. Chen, and S. Khanna, *Polynomial pass lower bounds for graph streaming algorithms*, in Symposium on Theory of Computing (STOC), 2019.

[4] S. Assadi, G. Kol, R. R. Saxena, and H. Yu, *Multi-Pass Graph Streaming Lower Bounds for Cycle Counting, MAX-CUT, Matching Size, and Other Problems*, in Foundations of Computer Science (FOCS), 2020.

[5] S. Assadi and V. N, *Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma*, in Symposium on Theory of Computing (STOC), 2021.

[6] P. Austrin, *Balanced Max 2-SAT might not be the hardest*, in Symposium on Theory of Computing (STOC), 2007.

[7] ———, *Towards sharp inapproximability for any 2-CSP*, SIAM Journal on Computing, (2010).

[8] S. K. Bera and A. Chakrabarti, *Towards Tighter Space Bounds for Counting Triangles and Other Substructures in Graph Streams*, in Symposium on Theoretical Aspects of Computer Science (STACS), 2017.

[9] A. Bernstein, *Improved Bounds for Matching in Random-Order Streams*, Theory of Computing Systems, (2023).

[10] A. Bhaskara, S. Daruki, and S. Venkatasubramanian, *Sublinear Algorithms for MAXCUT and Correlation Clustering*, in International Colloquium on Automata, Languages, and Programming (ICALP), 2018.

[11] J. Boyland, M. Hwang, T. Prasad, N. Singer, and S. Velusamy, *On sketching approximations for symmetric Boolean CSPs*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), 2022.

[12] M. Braverman, S. Garg, Q. Li, S. Wang, D. P. Woodruff, and J. Zhang, *A New Information Complexity Measure for Multi-pass Streaming with Applications*, in Symposium on Theory of Computing (STOC), 2024.

[13] V. Braverman, S. Chestnut, R. Krauthgamer, Y. Li, D. Woodruff, and L. Yang, *Matrix Norms in Data Streams: Faster, Multi-Pass and Row-Order*, in International Conference on Machine Learning (ICML), 2018.

[14] N. Buchbinder, M. Feldman, J. Seffi, and R. Schwartz, *A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization*, SIAM Journal on Computing, (2015).

[15] A. A. Bulatov, *A dichotomy theorem for nonuniform csps*, in Foundations of Computer Science (FOCS), 2017.

[16] K. Censor-Hillel, R. Levy, and H. Shachnai, *Fast Distributed Approximation for Max-Cut*, in Algorithms Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS), 2017.

[17] L. Chen, G. Kol, D. Paramonov, R. Saxena, Z. Song, and H. Yu, *Towards Multi-Pass Streaming Lower Bounds for Optimal Approximation of Max-Cut*, in Symposium on Discrete Algorithms (SODA), 2023.

[18] L. Chen, G. Kol, D. Paramonov, R. R. Saxena, Z. Song, and H. Yu, *Almost optimal super-constant-pass streaming lower bounds for reachability*, in Symposium on Theory of Computing (STOC), 2021.

[19] ———, *Near-optimal two-pass streaming algorithm for sampling random walks over directed graphs*, in International Colloquium on Automata, Languages, and Programming (ICALP), 2021.

[20] C.-N. Chou, A. Golovnev, A. Shahrasbi, M. Sudan, and S. Velusamy, *Sketching Approximability of (Weak) Monarchy Predicates*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), 2022.

[21] C.-N. Chou, A. Golovnev, M. Sudan, A. Velingker, and S. Velusamy, *Linear Space Streaming Lower Bounds for Approximating CSPs*, in Symposium on Theory of Computing (STOC), 2022.

[22] C.-N. Chou, A. Golovnev, M. Sudan, and S. Velusamy, *Sketching Approximability of All Finite CSPs*, Journal of the ACM, (2024).

[23] C.-N. Chou, A. Golovnev, and S. Velusamy, *Optimal Streaming Approximations for all Boolean Max-2CSPs and Max-kSAT*, in Foundations of Computer Science (FOCS), 2020.

[24] C. Demetrescu, I. Finocchi, and A. Ribichini, *Trading off space for passes in graph streaming problems*, ACM Transactions on Algorithms, (2010).

[25] A. Farhadi, M. T. Hajiaghayi, T. Mah, A. Rao, and R. A. Rossi, *Approximate Maximum Matching in Random Streams*, in Symposium on Discrete Algorithms (SODA), 2019.

[26] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory*, SIAM Journal on Computing, (1998).
[27] S. Guha and A. McGregor, *Tight Lower Bounds for Multi-pass Stream Computation Via Pass Elimination*, in International Colloquium on Automata, Languages, and Programming (ICALP), 2008.
[28] V. Guruswami and K. Onak, *Superlinear Lower Bounds for Multipass Graph Processing*, Algorithmica, (2016).
[29] V. Guruswami, A. Velingker, and S. Velusamy, *Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), 2017.
[30] R. Jayaram and D. P. Woodruff, *Towards Optimal Moment Estimation in Streaming and Distributed Models*, ACM Transactions on Algorithms, (2023).
[31] S. Kale and S. Tirodkar, *Maximum Matching in Two, Three, and a Few More Passes Over Graph Streams*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), 2017.
[32] J. Kallaugher and O. Parekh, *The Quantum and Classical Streaming Complexity of Quantum and Classical Max-Cut*, in Foundations of Computer Science (FOCS), 2022.
[33] J. Kallaugher, O. Parekh, and N. Voronova, *Exponential Quantum Space Advantage for Approximating Maximum Directed Cut in the Streaming Model*, in Symposium on Theory of Computing (STOC), 2024.
[34] M. Kapralov, S. Khanna, and M. Sudan, *Approximating matching size from random streams*, in Symposium on Discrete Algorithms (SODA), 2014.
[35] ———, *Streaming lower bounds for approximating MAX-CUT*, in Symposium on Discrete Algorithms (SODA), 2015.
[36] M. Kapralov, S. Khanna, M. Sudan, and A. Velingker, $(1+\omega(1))$*-approximation to MAX-CUT requires linear space*, in Symposium on Discrete Algorithms (SODA), 2017.
[37] M. Kapralov and D. Krachun, *An optimal space lower bound for approximating MAX-CUT*, in Symposium on Theory of Computing (STOC), 2019.
[38] S. Khot, *On the unique games conjecture (invited survey)*, in Conference on Computational Complexity (CCC), 2010.
[39] D. Kogan and R. Krauthgamer, *Sketching cuts in graphs and hypergraphs*, in Innovations in Theoretical Computer Science (ITCS), 2015.
[40] G. Kol, D. Paramonov, R. R. Saxena, and H. Yu, *Characterizing the Multi-Pass Streaming Complexity for Solving Boolean CSPs Exactly*, in Innovations in Theoretical Computer Science (ITCS), 2023.
[41] C. Konrad, F. Magniez, and C. Mathieu, *Maximum Matching in Semi-streaming with Few Passes*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), 2012.
[42] F. Kuhn, *The Price of Locality: Exploring the Complexity of Distributed Coordination Primitives*.
[43] F. Kuhn, T. Moscibroda, and R. Wattenhofer, *The price of being near-sighted*, in Symposium on Discrete Algorithms (SODA), 2006.
[44] M. Monemizadeh, S. Muthukrishnan, P. Peng, and C. Sohler, *Testable Bounded Degree Graph Properties Are Random Order Streamable*, in International Colloquium on Automata, Languages, and Programming (ICALP), 2017.
[45] E. Mossel, *Gaussian bounds for noise correlation of functions*, in Geometric and Functional Analysis, 2010.
[46] M. Parnas and D. Ron, *Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms*, Theoretical Computer Science, (2007).
[47] P. Peng and C. Sohler, *Estimating Graph Parameters from Random Order Streams*, in Symposium on Discrete Algorithms (SODA), 2018.
[48] P. Raghavendra, *Optimal algorithms and inapproximability results for every CSP?*, in Symposium on Theory of Computing (STOC), 2008.
[49] R. R. Saxena, N. Singer, M. Sudan, and S. Velusamy, *Improved streaming algorithms for Maximum Directed Cut via smoothed snapshots*, in Foundations of Computer Science (FOCS), 2023.
[50] R. R. Saxena, N. G. Singer, M. Sudan, and S. Velusamy, *Streaming complexity of CSPs with randomly ordered constraints*, in Symposium on Discrete Algorithms (SODA), 2023.
[51] T. J. Schaefer, *The complexity of satisfiability problems*, in Symposium on Theory of Computing (STOC), 1978.
[52] N. Singer, M. Sudan, and S. Velusamy, *Streaming approximation resistance of every ordering CSP*, Computational Complexity, (2024).
[53] N. G. Singer, *Oblivious algorithms for the Max-kAND problem*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), 2023.
[54] L. Trevisan, *Parallel Approximation Algorithms by Positive Linear Programming*, Algorithmica, (1998).
[55] D. Zhuk, *A proof of the CSP dichotomy conjecture*, Journal of the ACM (JACM), (2020).