Electronic Colloquium on Computational Complexity, Report No. 144 (2022)

Streaming beyond sketching for Maximum Directed Cut

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

Abstract

We give an $\widetilde{O}(\sqrt{n})$-space single-pass 0.483-approximation streaming algorithm for estimating the maximum directed cut size (Max-DICUT) in a directed graph on n vertices. This improves over an $O(\log n)$-space $4/9 < 0.45$ approximation algorithm due to Chou, Golovnev, Velusamy (FOCS 2020), which was known to be optimal for $o(\sqrt{n})$-space algorithms.

Max-DICUT is a special case of a *constraint satisfaction problem* (CSP). In this broader context, our work gives the first CSP for which algorithms with $\widetilde{O}(\sqrt{n})$ space can provably outperform $o(\sqrt{n})$-space algorithms on general instances. Previously, this was shown in the restricted case of bounded-degree graphs in a previous work of the authors (SODA 2023). Prior to that work, the only algorithms for *any* CSP were based on generalizations of the $O(\log n)$-space algorithm for Max-DICUT, and were in particular so-called "sketching" algorithms. Our work demonstrates that more sophisticated streaming algorithms can outperform these algorithms even on general instances.

Our algorithm constructs a "snapshot" of the graph and then applies a result of Feige and Jozeph (Algorithmica, 2015) to approximately estimate the Max-DICUT value from this snapshot. Constructing this snapshot is easy for bounded-degree graphs and the main contribution of our work is to construct this snapshot in the general setting. This involves some delicate sampling methods as well as a host of "continuity" results on the Max-DICUT behaviour in graphs.

*Microsoft Research. Email: `raghuvansh.saxena@gmail.com`

†Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. Supported by an NSF Graduate Research Fellowship (Award DGE2140739). Email: `ngsinger@cs.cmu.edu`.

‡School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts, USA. Supported in part by a Simons Investigator Award and NSF Award CCF 2152413. Email: `madhu@cs.harvard.edu`.

§School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts, USA. Supported in part by a Google Ph.D. Fellowship, a Simons Investigator Award to Madhu Sudan, and NSF Award CCF 2152413. Email: `svelusamy@g.harvard.edu`.

ISSN 1433-8092

Contents

1 Introduction

We consider approximating the directed cut value of a directed graph by a streaming algorithm presented with a stream of edges in an arbitrary (worst-case) order. Our result is a single-pass algorithm using $\widetilde{O}(\sqrt{n})$-space that gives a .483 approximation algorithm. In what follows we explain the background of this problem, the significance of the result, and the techniques used to achieve this result.

1.1 Background

We begin by defining the maximum directed cut (Max-DICUT) problem in a directed graph $\mathcal{G}$. (These definitions will all be informal; see Section 2 for formal definitions.) Given a graph $\mathcal{G}$ on n vertices, labeled $1, \ldots, n$, *cut* of $\mathcal{G}$ is a binary string $\mathbf{x} \in \{0,1\}^n$, assigning a bit to every vertex in $\mathcal{G}$. We say $\mathbf{x}$ *cuts* a directed edge (u, v) if $x_u = 1$ and $x_v = 0$. (Note the asymmetry between u and v.) The *value* $\mathsf{val}_{\mathcal{G}}(\mathbf{x})$ of a cut $\mathbf{x}$ is the total fraction of edges it cuts, and the *value* $\mathsf{val}_{\mathcal{G}}$ of $\mathcal{G}$ is the maximum value of any cut. A uniformly random cut has value $\frac{1}{4}$ in expectation, so every graph has value at least $\frac{1}{4}$.

We consider *streaming* algorithms for the problem of estimating the Max-DICUT value $\mathsf{val}_{\mathcal{G}}$ of a directed graph $\mathcal{G}$, given a stream $\boldsymbol{\sigma} = (e_1 = (u_1, v_1), \ldots, e_m = (u_m, v_m))$ of the graph's edges in arbitrary order. We say an algorithm is an *α-approximation* for the Max-DICUT problem if its output $\widehat{v}$ satisfies $\alpha \cdot \mathsf{val}_{\mathcal{G}} \leq \widehat{v} \leq \mathsf{val}_{\mathcal{G}}$ (with high probability). We say an algorithm is a *space-$s(n)$ streaming algorithm* (where n is the number of vertices in $\mathcal{G}$) if it reads the stream of edges $\boldsymbol{\sigma}$ in sequential order and uses $s(n)$ space.

The Max-DICUT problem is one example of a so-called *constraint satisfaction problem (CSP)*. We omit a full definition as we do not require it, but these problems are basically defined by two things: (1) a "global" space of allowed "assignments" to "variables" and (2) a collection of "local" constraints, each of which specifies allowed values for a small subset of variables. For Max-DICUT, assignments are cuts, variables are vertices, and constraints are edges; we will use these terms interchangeably. The "symmetric version" of Max-DICUT is another CSP called *maximum cut* (Max-CUT), in which a cut $\mathbf{x}$ cuts an edge (u, v) if $x_u \neq x_v$; we mention it here as it serves a useful point of comparison for Max-DICUT.

1.2 Recent work

Over the last decade, there has been extensive work on the approximability of various CSPs in various streaming models [KK15, KKS14, GVV17, KKSV17, GT19, KK19, CGV20, CGSV21, SSV21, CGS+22b, BHP+22, CGS+22a, SSSV23]; see also the surveys [Sin22, Sud22].

Max-DICUT has emerged as the central benchmark for algorithms among CSPs in the streaming setting. It was the first problem shown to admit a non-trivial approximation in sublinear (in n) space in the work of Guruswami, Velingker, and Velusamy [GVV17]. Subsequent work of Chou, Golovnev, and Velusamy [CGV20] gave an improved algorithm for Max-DICUT along with a tight bound on the approximability — pinning the approximability of Max-DICUT for $o(\sqrt{n})$-space streaming at $\frac{4}{9}$.

Theorem 1.1 ([CGV20]). *For every $\epsilon > 0$, there is a streaming algorithm (in fact, a linear sketching algorithm) which $(4/9 - \epsilon)$-approximates the* Max-DICUT *value of a graph in $O_\epsilon(\log n)$ space. Conversely, every $(4/9 + \epsilon)$-approximation streaming algorithm for* Max-DICUT *uses $\Omega_\epsilon(\sqrt{n})$ space.*

This result is part of a broader landscape for $o(\sqrt{n})$-space streaming complexity of CSPs. In particular, Chou, Golovnev, Sudan, and Velusamy [CGSV21] proved a *dichotomy theorem* for all finite CSPs. The understanding of Max-DICUT plays a central role in their results. In particular, they generalize the Max-DICUT algorithm of [CGV20] to all CSPs. Their lower bounds also generalize the lower bounds from [CGV20] with some notions ("padded one-wise independent problems") that are direct abstractions of Max-DICUT and share tight lower bounds.

A *sketching algorithm* is a kind of streaming algorithm which chooses its output based on a short "sketch" of its input, where these sketches have the composability property that the sketch of a concatenation of two streams can be obtained from the sketches of the two components. (See [CGSV21] for a formal definition.) The algorithms of [GVV17, CGV20] for Max-DICUT were sketching algorithms and so were their generalizations in [CGSV21]. Our work is primarily motivated by the quest to look beyond sketching algorithms for streaming CSPs. One (over-)optimistic goal might be to find algorithms that outperform the current sketching algorithm for all CSPs.

But this hope is not achievable in sublinear space. For a wide class of CSPs, including Max-CUT, there are recent $o(n)$-space lower bounds ruling out all nontrivial approximations [KK19, CGS$^+$22b].[1][2] Thus to make advances one has to restrict the problems considered and in this work we focus on the simplest remaining problem after Max-CUT, namely, Max-DICUT. For Max-DICUT, till this work and a recent related work by the authors [SSSV23] it was conceivable that there were no improvements possible in $o(n)$ space. But at the same time the above mentioned lower bounds did not extend to this setting and it was unclear whether this was due to a limitation of the lower bounds techniques or if better algorithms exist.

In a previous work [SSSV23], the authors gave some evidence for the possibility that better algorithms for Max-DICUT do indeed exist. To be precise, recall that the sketching algorithm of [CGV20] is a $\frac{4}{9} \approx 0.444$-approximation, which uses $O(\log n)$ space and is optimal among $o(\sqrt{n})$-space streaming algorithms (Theorem 1.1). In [SSSV23] we proved that for Max-DICUT, the algorithm of [CGV20] *can* be beaten in a number of restricted models such as when the input stream is *randomly* (instead of adversarially) ordered, or the graph has constant max-degree. In particular:

Theorem 1.2 ([SSSV23])**.** *For every $d \in \mathbb{N}$, there is a streaming algorithm which* 0.483-*approximates the* Max-DICUT *value of a graph with maximum degree d in $\widetilde{O}_d(\sqrt{n})$ space.*

Theorem 1.2 does not answer the question of whether the Max-DICUT algorithm of [CGV20] can be beaten on *general* graphs in $\o(n)$ space. Indeed, it could be considered evidence only that even-more-sophisticated lower bound techniques are necessary to rule out such algorithms. We further discuss why we believe that Theorem 1.2 was far from an answer to this question in Section 1.4 below.

1.3 Main result

Our main theorem gives an algorithm that uses slightly more than $\sqrt{n}$ space and outperforms the algorithm of [CGV20]:

Theorem 1.3 (Main theorem)**.** *There is a streaming algorithm which* 0.483-*approximates the* Max-DICUT *value of an arbitrary graph in $\widetilde{O}(\sqrt{n})$ space.*

This theorem is proven in a more precise formulation as Lemma 4.2 below.

[1] $o(n)$ space is tight up to logarithmic factors because randomly sparsifying down to $O(n/\epsilon^2)$ constraints gives $(1-\epsilon)$-approximations.

[2] The condition for inapproximability given in [CGS$^+$22b] for a predicate $f : \mathbb{Z}_q^k \to \{0,1\}$ is termed "width", and states that f's support contains some translate of the diagonal $\{(a,\ldots,a) : a \in \mathbb{Z}_q^k\}$. More broadly, the strongest known hardness results for CSPs (e.g., also in [CGSV21]) seem to rely on "niceness" properties of the support of f.

Like the previous algorithms for $\mathsf{Max\text{-}DICUT}$ [GVV17, CGV20, SSSV23], our algorithm relies crucially on the notion of the *bias* of a vertex in a directed graph. Given a directed graph $\mathcal{G}$ and a (positive-degree) vertex v, we define the bias $\mathsf{bias}_{\mathcal{G}}(v)$ of v as the difference between its out- and in-degrees divided by their sum. This is a number between -1 and 1 which represents v's "preference" for being assigned 1 vs. 0.

Earlier papers working in the $O(\log n)$-space setting [GVV17, CGV20] measured a quantity called the *total bias* of $\mathcal{G}$, namely $\mathsf{bias}_{\mathcal{G}} = \frac{1}{2m}\sum_{v=1}^{n} \mathsf{deg}_{\mathcal{G}}(v)|\mathsf{bias}_{\mathcal{G}}(v)|$, using standard ℓ_1-norm sketching algorithms and used this as a proxy for the value of $\mathcal{G}$.[3] The current work relies on measuring a so-called "snapshot" of the graph, which roughly says how many edges go from vertices of bias $\approx s$ to vertices of bias $\approx t$, for every choice of $s, t \in [-1, 1]$. Plugging this snapshot into an algorithm due to Feige and Jozeph [FJ15] gives us our estimator for the $\mathsf{Max\text{-}DICUT}$ value. Thus the novelty of our algorithm is in producing the snapshot in roughly $\sqrt{n}$ space.

We remark that our algorithm is not a *sketching* algorithm, unlike all previously known algorithms for CSPs (in the standard model). We do not currently know that sketching algorithms cannot match the performance of our algorithm, but we also do not know a sketching algorithm that can.

1.4 Beyond bounded-degree instances

As mentioned above, our prior work [SSSV23] already achieves our main theorem for the special case of bounded-degree graphs. In this section we argue why the extension to the general case is non-trivial and important.

We start with considering the previous works of [GVV17, CGV20, CGSV21]. The algorithms in all these works use powerful norm estimation algorithms as black boxes. If one were to consider the simpler case of their problems in the bounded-degree setting, these algorithms could have been implemented without reliance on these subroutines. The $\mathsf{Max\text{-}DICUT}$ algorithms only need an estimate of the absolute value of "bias times the degree" for a random vertex, and this could be estimated by simply picking a random sample of the vertices and computing their bias and degree as the stream passes by. For general CSPs (even on non-Boolean domains) also such a process would suffice, and this would not only simplify the algorithms significantly, it even would achieve a space bound of $O(\log n)$ which is better than the current bounds given in [CGSV21] for general CSPs.

Digging deeper into this analogy one can consider ℓ_p norm estimation problems themselves. For this class of problems also one can define a bounded-degree version of the problem — where one is trying to compute the ℓ_p norm of a vector in $\{-C, \ldots, C\}^n$ in the turnstile update model. In this bounded-degree setting, the ℓ_p norm can be trivially computed by randomly sampling an $O_C(1)$-sized subset of the coordinates and maintaining their values. Thus ℓ_p norms can be estimated in $O(\log n)$ space for every p in this bounded-degree setting, whereas in the general case it is well-known that ℓ_p norm estimation requires polynomial in n space for $p > 2$.

Thus, the bounded-degree setting can be vastly easier to solve and results in this setting may best be viewed as a proof of concept — though even this "proof of concept" may be misleading, as exemplified by the ℓ_p norm estimation problem.

Turning to our specific goal — that of computing snapshots of a graph in $\widetilde{O}(\sqrt{n})$ time — our prior work [SSSV23] again manages to estimate this snapshot in the bounded-degree setting by

[3] In other words, $\mathsf{bias}_{\mathcal{G}}$ averages the absolute difference between out- and in-degree over vertices in the graph. Intuitively, the larger $\mathsf{bias}_{\mathcal{G}}$ is, the more "opinionated" each vertex is and therefore the easier it is to find a high-value assignment; this is made quantitative in [GVV17, CGV20]. Note also that $\mathsf{bias}_{\mathcal{G}}$ is the ℓ_1 norm of the vector of differences of out- and in-degrees, and each edge in the stream adds 1 to one entry and subtracts 1 from another.

sampling $\widetilde{O}(\sqrt{n})$ vertices and maintaining the bias of the sampled vertices as well as the induced subgraph on these vertices. We discuss why this is reasonable in the bounded-degree setting in the following subsection. But such a simple algorithm is definitely not going to work in the general setting! In particular, computing a good estimate of the snapshot is at least as hard as computing the ℓ_1 norm of a vector in the turnstile model with unit updates. Indeed, "snapshot" estimation seems to be a "higher-level" challenge than simple norm estimation and roughly requires computing some "two-wise" marginals of the graph updates, whereas bias corresponded to "one-wise" marginals. Black-box use of norm-estimation algorithms no longer seems to suffice to solve these "two-wise" marginal problems, which seem to need new algorithmic ideas. We feel this class of problems and the ideas used here to deal with them may be of even broader interest than the application to Max-DICUT.

1.5 Techniques: Streaming estimation of "snapshots"

In this section, we dive into the specific challenges arising in computing "snapshots" of graphs in the general setting. We also begin to describe our algorithm; we aim to progressively narrate and motivate its development as there are a number of technical issues which arise. The algorithm proper is described completely in Section 4.

Roughly, the Feige-Jozeph algorithm [FJ15] can be viewed as a reduction from the problem of approximating the Max-DICUT value of a graph $\mathcal{G}$ to a problem of the form,

> Given a graph $\mathcal{G}$ and fixed $s < s'$, $t < t'$, estimate the number of edges $(u, v) \in E(\mathcal{G})$ such that $s \leq \mathsf{bias}_{\mathcal{G}}(u) < s'$ and $t \leq \mathsf{bias}_{\mathcal{G}}(v) < t'$.

More precisely, [FJ15] gives us some fixed partition $t_0 = -1 < \cdots < t_\ell = 1$ of the space of possible biases. Given a graph $\mathcal{G}$, we refer to all vertices $v \in V(\mathcal{G})$ with bias in the interval $[t_{i-1}, t_i)$ as *bias class* i. For a graph $\mathcal{G}$, we let $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ denote the $\ell \times \ell$ matrix whose (i, j)-th entry counts the fraction of edges in $\mathcal{G}$ from bias class i to bias class j. We refer to this matrix informally as a "bias snapshot" of $\mathcal{G}$. We can use the algorithm of [FJ15] as a black box to approximate the Max-DICUT value of a graph $\mathcal{G}$ up to a factor 0.483, given an estimate for this snapshot matrix in ℓ_1-norm (see Corollary 3.4 below).

1.5.1 Algorithms based on vertex-subsampling

Setting aside the streaming model momentarily, the most natural way to build a snapshot for general graphs is to sample $O(\log n)$ edges from $\mathcal{G}$ independently and uniformly at random, and measure the biases of their two endpoints. Using these computed values, one can then estimate the entries of the matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ and standard concentration inequalities will yield the required approximation to Max-DICUT. Unfortunately, in the streaming setting when the edges are adversarially ordered there is no obvious way to implement the above sampling procedure since by the time the "random" edge appears in our stream, most of the edges incident to the endpoints might have already appeared, and thus, we may not know their biases.[4]

Subsampling vertices independently. Another approach one might try is to track the bias of a subset S of vertices from the beginning of the stream and restrict attention to the edges in

[4]As observed in [SSSV23], when the edges in the stream are *randomly* ordered this simple setup does give an algorithm: One can simply record the first $O(\log n)$ edges in the stream and then observe their biases over the remainder of the stream.

the subgraph induced by the vertices in S. If each vertex of $\mathcal{G}$ is included in S with probability, say, $\rho = \sqrt{\frac{\log m}{m}}$ so that the probability that an edge in $\mathcal{G}$ is in the induced subgraph is $\rho^2 = \frac{\log m}{m}$, sampling this way has the nice property that (like in the "ideal" case of directly sampling edges) the number of edges in the induced subgraph is $O(\log m)$. Moreover, as one only needs to store the bias of the vertices in S which are *non-isolated* in $\mathcal{G}$ and there are at most $2m$ non-isolated vertices in $\mathcal{G}$ (each of which is included in S with probability ρ), sampling this way requires space at most $2m \cdot \rho = \widetilde{O}(\sqrt{m})$. As $m \leq \widetilde{O}_\epsilon(n)$, this is also $\widetilde{O}(\sqrt{n})$ and so fits within the space bound.

Unfortunately, sampling this way also means that the edges in the induced subgraph are no longer independent. Indeed, consider two edges e_1 and e_2 that share a common vertex v. If e_1 is in the induced subgraph, then v must be in S, increasing the chance that e_2 is also in the induced subgraph, and breaking the independence. Without the independence, we can no longer apply concentration inequalities and get an estimate for the $\mathsf{Max\text{-}DICUT}$ value of $\mathcal{G}$. However, not all hope is lost, as two edges are independent only if they share a common endpoint, and *e.g.*, if the graph has small maximum degree, then we can control the amount of dependence between edges and still make the argument successful. In fact, this is exactly what was done in the algorithm presented in [SSSV23] for graphs with small maximum degree (Theorem 1.2 above).

To be precise, the argument for correctness of the estimate in the bounded-degree case uses a form of Chebyshev's inequality (see Corollary 2.8). If we sample each vertex with probability p in a graph with maximum degree d, and our estimate an entry of $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ will be within ϵm of the true value except with probability $O(d/p^2\epsilon^2 m)$. So, the larger the maximum-degree d, the larger p must be to ensure correctness; $\omega(\sqrt{n})$ space is required even for $d = n^{0.01}$.

Towards general graphs. How can we hope to extend this procedure to general graphs? Firstly, we observe that a standard sparsification argument shows that $\mathsf{Max\text{-}DICUT}$ algorithms for graphs with $O(n/\epsilon^2)$ edges lift to $\mathsf{Max\text{-}DICUT}$ algorithms for *all* graphs, with ϵ loss in the approximation factor (see Lemma 2.9 below). This lets us reduce to only considering graphs with $O(n)$ edges.

Our goal is to build on the vertex sampling approach described above, and show that it can be extended to work for general graphs. For this, we first observe that the approach above works as is even if the graph does not have small maximum degree, as long as the degree of non-isolated vertices in $\mathcal{G}$ are "roughly the same". Indeed, if all the positive-degree vertices in $\mathcal{G}$ have degree, say, between d and $2d$, one can subsample the edges of the graph independently with probability C/d for some large constant C; the resulting graph will, with high probability, have maximum degree $O(C)$ and preserve the bias matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ up to additive (ℓ_1-norm) errors. Then, one can apply the bounded-degree case from [SSSV23] and get a required approximation for the $\mathsf{Max\text{-}DICUT}$ of $\mathcal{G}$.

The more challenging (and general) case is when the graph contains vertices with a wide range of different degrees. Roughly, the plan is to fix an ordered partition $\mathbf{d} = (d_1, \ldots, d_k)$ of the possible degrees in the graph $\mathcal{G}$. Then, for each $a \in \{1, \ldots, k\}$ we aim to sample a set of vertices whose degree is "roughly" d_a. To do so, we first subsample a graph $\mathcal{G}_a \subseteq \mathcal{G}$ by including each edge (randomly, independently) with probability $q_a = C/d_a$ for some large constant C. Note that with high probability, vertices with degree roughly C in $\mathcal{G}_a$ will correspond to vertices with degree roughly d_a in $\mathcal{G}$. The next step is to subsample the positive-degree vertices in $\mathcal{G}_a$ by including each vertex (randomly, independently) with an appropriately chosen probability p_a, and then use information about these vertices to try and estimate our snapshot.

The bias-degree array. It is no longer clear how we can directly estimate the bias matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$. Instead, we first focus on estimating the more complex array $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$ whose

(a, b, i, j)-th entry contains the fraction of edges in $\mathcal{G}$ that go from vertices in bias class i and degree class[5] a to vertices in bias class j and degree class b. Note that this array is more granular than the matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ which we actually want to estimate — in particular, the matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ can be computed from the array $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$ by "projecting" it to its third and fourth coordinates — but this change is reasonable in our setting because the parameters for subsampling edges and vertices are heavily dependent on the degrees of vertices of interest. We will abbreviate $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$ for convenience and focus on computing (actually, estimating) A via a "subsampling strategy" like the above.

1.5.2 Estimating the array using edge- and vertex-subsampling

Fix some degree partition $\mathbf{d} = (d_1, \ldots, d_k)$ in which $d_{a+1}/d_a = O(1)$ for all a; we use $d_{a+1} = 2d_a$ (note that $k = O(\log n)$). We think of each $a \in [k]$ as a *layer* which targets vertices in degree class a, i.e., vertices with degree in the range $(2^{a-1}, 2^a]$, and makes sure that these vertices have small maximum degree in the subsampled graph $\mathcal{G}_a$. This is done by sampling each edge in $\mathcal{G}$ independently with probability roughly 2^{-a}, and considering the graph $\mathcal{G}_a$ consisting of the sampled edges. Then, we subsample a set S_a of positive-degree vertices in $\mathcal{G}_a$.

Given an edge which is somehow sampled in this kind of procedure, we will want to know the bias (and degree) of its endpoints in order to add it into the appropriate entry of A. In general, when a vertex shows up in the stream for a subsampled graph $\mathcal{G}_a$ many of its incident edges in the base graph $\mathcal{G}$ may have slipped by! However, we can use the bias and degree of a vertex $v \in S_a$ *within the subgraph* $\mathcal{G}_a$ as an estimate for the bias and degree, respectively, in the base graph $\mathcal{G}$, as long as v's degree is "decently high". We mostly ignore this issue in this subsection, but it is quite important in the analysis and we discuss it in the next subsection.

Estimating edges within each layer. Consider the simple strategy where we just store the subgraph induced on S_a within the graph $\mathcal{G}_a$. We can look at these induced subgraphs and — modulo the issue of estimating the bias and degree of sampled vertices — estimate, for all $a \in [k]$, the entries of A corresponding to the edges going from vertices of degree class a to degree class a with bias classes i and j. However, this is only a small subset of the entries of A which we need to estimate. Given $a \neq b \in [k]$, how can we estimate the "cross edges" between a and b?

Modifications for estimating cross edges. The above approach does not let us estimate the number of cross edges directly. To see why, fix $a < b \in [k]$ and observe that vertices in degree class b are expected to have a high degree in layer a (as they are expected to have 2^b edges and we subsample with probability 2^{-a}) while the vertices in degree class a are expected to have almost no edges in layer b. The former means that we cannot estimate the cross edges from the graph $\mathcal{G}_a$ as it has high maximum degree, while the latter means that we cannot estimate the cross edges from the graph $\mathcal{G}_b$, as we lose almost all information about the biases and degrees of layer a.

In order to estimate the cross edges, we strengthen the sampling algorithm in the following two ways:

1. Firstly, when looking at the layer $\mathcal{G}_a$, in addition to the edges in the subgraph induced by the set of vertices S_a sampled in this layer, we also remember *all* edges in $\mathcal{G}_a$ that are incident on a vertex in S_a whose degree class is a. As the algorithm does not know the degree class of any vertex *a priori*, this is actually implemented indirectly, namely, by remembering $\log^{O(1)} n$ many edges from *all* vertices in S_a. This "cutoff" number of edges is small enough to not

[5]Similar to how we defined bias classes, degree class a is the set of vertices with degree in between d_{a-1} and d_a.

increase the space used by a lot, but also large enough to allow all the edges of vertices in degree class a to be remembered (as these vertices are expected to have low degree in $\mathcal{G}_a$).

2. Secondly, observe that as the layer a increases, the number of edges in $\mathcal{G}_a$ decreases. By the calculation in Section 1.5.1, this means that the amount of memory required to remember the biases and degrees of the vertices in S_a also decreases. As the number k of layers is small, we actually do not need this memory to decrease, and can actually afford to give all the layers the same memory as the first layer, namely $\widetilde{O}(\sqrt{n})$. By appropriately changing the parameters in the "higher" layers, this extra memory allows us to remember the biases of an even larger set S_a, which will be useful in estimating the cross edges precisely.

How to estimate cross edges? We claim that, with these modifications, one can estimate cross edges between layers a and b by looking at the graph $\mathcal{G}_a$ and counting the number of edges in this graph that go from vertices in S_a to vertices in S_b. To see why this works, we consider $a = 1$ and two cases for b:

- **When $2^b \geq \sqrt{n}$:** In this case, we claim that S_b is sufficiently large as to contain all the vertices in $\mathcal{G}$ with degree class b. Indeed, as the the degree of such vertices is around $2^b \geq \sqrt{n}$ and the total number of edges in $\mathcal{G}$ is $m \leq \widetilde{O}_\epsilon(n)$, the number of vertices in $\mathcal{G}$ with degree class b cannot be much larger than $\sqrt{n}$. As explained in Item 2, this means that all these vertices will be in S_b.

  Given the fact that S_b has all these vertices and we have Item 1, whether or not a cross edge $e = (u, v)$, where u has degree class a and v has degree class b, is counted depends only on whether or not $u \in S_a$ and whether or not $e \in \mathcal{G}_a$. The latter is independent across all edges while the former has only a small amount of dependence, as the vertices in degree class a have low degree in $\mathcal{G}_a$, and does not harm the concentration inequalities too much.

- **When $2^b < \sqrt{n}$:** The argument above will not directly work in this case, as now whether or not a cross edge is counted also depends on whether or not $v \in S_b$. As the vertices in degree class b have high degree in $\mathcal{G}_a$, this creates a lot of dependencies (depending on 2^{b-a}) and breaks the concentration bounds.

  What saves us here is that in Items 1 and 2, we sample all edges in $\mathcal{G}_a$ that are incident on S_a and also remember extra vertices in S_b. Thus, the number of cross edges between a and b that are remembered in $\mathcal{G}_a$ is much larger than $O(\log m)$ (which was the number obtained in the bounded-degree case). Having a larger number of cross edges also means we can also afford to deviate by more without affecting the multiplicative guarantee, and this larger deviation will help us deal with the extra dependencies in this case.

1.5.3 Windows, smoothing, and continuity

Even with the modifications above, there is a major problem that we still have to overcome. This problem arises because we do not compute the degrees and biases of the vertices in $\mathcal{G}$ exactly, and instead *estimate* them from the sampled graphs $\mathcal{G}_a$. These estimates will always be slightly off, and this can wreak havoc in the analysis, as the following example shows.

Consider an edge containing a vertex v with degree 2^a so that it is at the "boundary" of degree classes a and $a + 1$. (One could also consider a vertex with bias t_i, which is at the boundary of bias classes i and $i + 1$, or a vertex that is at the boundary of both.) It is impossible to determine with high probability which entries of the estimate for A such an edge will contribute to — in some subsamples, the vertex could "appear to be" in degree class a and in others it could be in $a+1$. Thus, we will end up counting this vertex in *both* the classes, making our estimates off by a factor of 2 for these vertices! Note that the vertices that are not at the boundary will only be counted in one class, and we cannot simply divide by 2 throughout to make this problem go away.

The solution we come up with is to use windows: Instead of trying to estimate entries of A individually, we group them into "windows" and estimate the average of all the entries in the window instead. For example, when trying to estimate the degree class a, we instead take a windowing parameter w and estimate the average of all degree classes $\{a-w,\dots,a+w\}$[6]. This mellows the problem of overestimating by a factor of 2 and makes it manageable: Now, *every* entry is over-counted $2w+1$ times and the boundary entries, in the worst case, will be over-counted at most $2(w+1)+1$ times. As $\frac{2(w+1)+1}{2w+1} \approx 1$ for large w, this is still manageable.

However, we now need to show that we can rely on the algorithm of [FJ15] to compute a Max-DICUT value even if we only have windowed averages of A instead of A itself. This constitutes another major part of our analysis (see Section 3) where we essentially show that the value of Max-DICUT is "continuous" in the sense one can work with the weak "smoothed" estimates resulting from windows without losing out on the overall approximation factor. This part may be of independent interest.

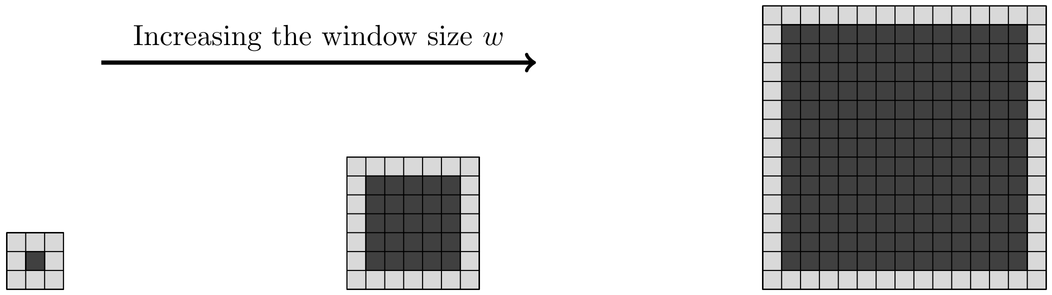

Figure 1: A depiction of how larger windows reduce the "borderline" effects (in two dimensions). As w becomes larger and larger, a $w \times w$ rectangle (dark gray) dominates its "boundary" (light gray) more and more. Geometrically, a rectangle is two-dimensional while its boundary is "essentially one-dimensional". However, for estimating the Max-DICUT value in a graph, smoothing over size-w windows for large w introduces errors from the use of "continuity" results (i.e., Lemma 3.13 below). The right choice of w strikes a balance between these two forces.

Brief outline of Section 3. We have a few key lemmas about the windowing operation. Our key "continuity" lemma, Lemma 3.13 below, states that this operation preserves the Max-DICUT value of the graph in a precise sense, which may be of independent interest. We prove Lemma 3.13 constructively in Section 6.5 below, by "blowing up" each vertex into many copies and slightly tweaking the biases and degrees of each.

Separately, we also define a pair of arrays, which we denote A^{-w} and A^{+w}, which capture the "worst lower bound" and "worst upper bound", respectively, for the smoothed array $A^{\sim w}$ when allowing "off by one" errors for which entries count. In other words, each entry of A^{-w} sums over a smaller surrounding window of width only $w-1$, while entries of A^{+w} sum over larger windows

[6]One important issue with this approach is we have to handle the "degenerate" cases where, e.g. $a < w$ so the set of allowed degree classes is smaller than $2w+1$. We correspondingly have to weight the entries in the matrix to equalize the contributions of different edges, and this introduces some more potential errors in the algorithm as these "weighting factors" for sampled edges can also be estimated incorrectly. We ignore these details in this introduction.

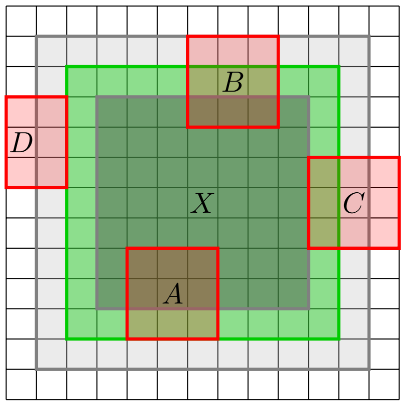

Figure 2: Consider estimating a (two-dimensional) matrix with "off-by-one" errors, wherein the mass of each entry may shift to one of 8 neighboring entries (red boxes). If we estimate an average over a window of size $w = 4$ in taxicab distance around an entry X (green rectangle): (i) "Outer" entries, such as the one marked D, beyond distance $w + 1 = 5$ from the center (light gray) can *never* contribute. (ii) "Inner" entries, such as the one marked A, within distance at most $w - 1 = 3$ from the center (dark gray) *always* contribute. (iii) "Borderline" entries, such as B or C, *may or may not contribute*, depending on the specific error pattern.

of width $w + 1$. Thus, we have $A^{-w} \leq A^{\sim w} \leq A^{+w}$ entrywise. In Lemma 3.20 below, we show that $\|A^{+w} - A^{-w}\|_1 \leq O(1/w)$ and thus it makes sense to only aim for an estimate between A^{-w} and A^{+w} (as opposed to aiming directly for $A^{\sim w}$). We call such an estimate a "weak estimate" of $A^{\sim w}$.

These lemmas in turn are the basis of our overall "reduction" lemma, Lemma 3.22, which gives a set of conditions on a "weak estimate" for $A^{\sim w}$ which are sufficient to guarantee that it can be plugged into the algorithm of Feige and Jozeph [FJ15] (with controlled loss in the approximation factor). This lemma is complemented by our key algorithmic correctness lemma, Lemma 4.4, which states that (under a few technical assumptions which hold essentially without loss of generality), we can produce such "weak estimates" from the sampled edges and vertices with high probability.

1.6 Future directions

Via the trivial reduction from Max-CUT, it is known that for all $\epsilon > 0$, streaming algorithms which $(\frac{1}{2} + \epsilon)$-approximate Max-DICUT require $\Omega_\epsilon(n)$ space (cf. [CGS+22b]). There are a number of interesting alternatives for what could happen between $\omega(\sqrt{n})$ and $o(n)$ space. Three main scenarios are:

1. "Algorithms beating Theorem 1.3": There are $(\frac{1}{2} - \epsilon)$-approximations in $\widetilde{O}(\sqrt{n})$ space.
2. "Lower bounds matching Theorem 1.3": Beating 0.483 requires $\widetilde{\Omega}(n)$ space.[7]
3. "Approximation vs. space tradeoff": Beating 0.483 can be achieved in $o(n)$ space, but $(\frac{1}{2} - \epsilon)$-approximations for $\epsilon > 0$ require arbitrarily close to $\Omega(n)$ space for arbitrarily small ϵ.

We include a few brief remarks on these possibilities and speculate on why something like the third alternative may be plausible. Firstly, we mention that on the algorithmic side, in the classical

[7]Actually, 0.483 is not *exactly* the best we can do; the correct constant characterizing the best performance of "snapshot-based algorithms", which we denote α_{FJ}, was bounded by Feige and Jozeph [FJ15] between 0.483 and 0.4899. See Lemma 3.3 below.

setting Trevisan [Tre98] analyzed the natural linear programming (LP) relaxation for Max-DICUT, and showed that this LP achieves a $\frac{1}{2}$-approximation (though he mentions the result was already folklore); Halperin and Zwick [HZ01] showed further that one can always take the optimal solutions of the LP to be *half-integral* and used this to give an alternative "combinatorial" interpretation of the LP.[8] Is there some hope for collecting enough information in the streaming setting (even in, say, $O(n^{0.99})$ space) to simulate one of these algorithms?

Secondly, we recall that the original streaming inapproximability result for the related problem Max-CUT, due to [KKS15], used the hardness of a two-player one-way communication problem called Boolean-Hidden-Matching (introduced in [GKK$^+$08]). Very informally, in this game, Alice receives some information about each vertex, and Bob receives some edges and information about each edge which supposedly corresponds to Alice's knowledge of its endpoints.[9] How much information does Alice have to send to Bob for him to tell whether his per-edge information is correct? $\Omega(\sqrt{n})$ space is sufficient because of the so-called "birthday argument": Alice can send her information for $\Omega(\sqrt{n})$ vertices, at which point it becomes likely that Bob will see $\Omega(1)$ edges which Alice has informed him about. Thus, the simplest hardness proofs for CSPs break down at $\Omega(\sqrt{n})$ space, and this is for precisely the same reason the above algorithm for Max-DICUT works with $\widetilde{O}(\sqrt{n})$ space!

The authors of [KKS15] also observed that a more general problem, called Boolean-Hidden-Hyper-Matching (introduced in [GKK$^+$08]), in which Bob instead receives t-uniform hyperedges, gives (suboptimal) hardness for Max-CUT in $o(n^{1-1/t})$ space. (This is because for Bob to see $\Omega(1)$ correctly labeled edges in the "birthday" protocol, Alice needs to send labels for each vertex with probability $n^{-1/t}$.) This was subsequently extended in [KK19] to optimal hardness for Max-CUT even in $o(n)$ space.

All this is to say that it is possible for there to be threshold at, for instance, $o(n^{2/3})$ space, where Max-DICUT's approximability changes. Indeed, there is a concrete reason to ask about this. Feige and Jozeph [FJ15] construct a pair of graphs, whose Max-DICUT values differ by a factor 0.4899, which cannot be distinguished by algorithms which only inspect edges and the biases of their endpoints. But, these graphs can easily be distinguished by algorithms which inspect paths of *length* 2 and the biases of their vertices. Seeing such paths, at least via the natural sampling procedure, requires storing $\Omega(n^{2/3})$ vertices.[10] So can this information be used to improve on the algorithm in Theorem 1.3 given $\widetilde{O}(n^{2/3})$ space? Conversely, can n-vertex "blowups" of the [FJ15] graphs be used to prove a 0.4899-inapproximability result in $o(n^{2/3})$ space — perhaps based on some communication problem such as Boolean-Hidden-Hyper-Matching?

Outline of the paper

In Section 2 we introduce some notation and review some background material. The main technical content of the paper is from Section 3 onwards, which can be divided into two independent steps. In the first step, we reduce approximating Max-DICUT (up to a factor 0.483) on a graph $\mathcal{G}$ to estimating a certain kind of "weak snapshot" of the graph $\mathcal{G}$. In the second step, we show how such

[8]Specifically, Halperin and Zwick [HZ01] proved that half-integrality of the LP by showing that it's equivalent to solving a maximum fractional independent set problem on the *line graph* of $\mathcal{G}$, which in turn reduces to a bipartite matching problem on a certain related graph. See [GVV17, Appendix A] for an alternate proof of the half-integrality result.

[9]In particular, Alice receives a cut (a bit labeling each vertex). Bob's edges form a random matching and his edge-labeling either encodes which edges cross Alice's cut, or is uniformly random.

[10]This is an interesting contrast with the random-ordering model, where these sorts of quantities seem to be estimable in roughly $O(\log n)$ space (cf. [SSSV23]).

a "snapshot" can be measured using a streaming algorithm, and use this to design an approximation algorithm for Max-DICUT.

The first step begins in Section 3. Here, we formally define objects such as the "bias matrix" and "bias-degree array" of a graph which capture our informal notions of snapshots. Then, we give a self-contained statement of the type of estimate we are looking to find in the algorithm. This section culminates in a lemma (Lemma 3.22) which "reduces" the Max-DICUT approximation task to a weak estimation problem for the bias-degree array. We state also various smaller lemmas which together imply Lemma 3.22. We postpone the proofs of the lemmas until Section 6 and move onto the algorithm.

The second step takes place in Sections 4 and 5. Specifically, in Section 4, we design an algorithm for "weak estimate" which fulfills the premises of Lemma 3.22; we also include various "wrapper" algorithms using this to actually estimate the Max-DICUT value of a graph, culminating in a proof of Theorem 1.3 (in its technical form Lemma 4.2). In Section 5 we prove our central correctness lemma (Lemma 4.4), which states roughly that the estimate produced by the algorithm fulfills the premises of Lemma 3.22, by analyzing all the errors that arise in the algorithm from sampling, estimating the biases and degrees of vertices, etc.

2 Preliminaries and notation

$[\ell]$ denotes the set of natural numbers $\{1, \ldots, \ell\}$. We use standard asymptotic notation $O(\cdot), o(\cdot)$, etc., with the convention that subscripts (e.g., $f(x, y) = O_y(g(x))$) denote arbitrary dependence in the implicit constant.

2.1 Matrices and arrays

For $\ell \in \mathbb{N}$, we let $\mathbb{M}^\ell \overset{\text{def}}{=} \mathbb{R}^{\ell \times \ell}$ denote the space of real $\ell \times \ell$ matrices, $\mathbb{M}^\ell_{\geq 0} \subseteq \mathbb{M}^\ell$ the space of matrices with nonnegative entries, and $\mathbb{M}^\ell_\Delta \subseteq \mathbb{M}^\ell$ matrices with nonnegative entries summing to 1. For $i, j \in [\ell]$, $M(i, j)$ denotes the (i, j)-th entry of M. Given two matrices $M, N \in \mathbb{M}^\ell$, we let $\|M - N\|_1$ and $\|M - N\|_\infty$ denote their entrywise 1- and ∞-norms, respectively, i.e.,

$$\|M - N\|_1 \overset{\text{def}}{=} \sum_{i,j=1}^{\ell} |M(i, j) - N(i, j)| \text{ and } \|M - N\|_\infty \overset{\text{def}}{=} \max_{i,j \in [\ell]} |M(i, j) - N(i, j)|.$$

For $k, \ell \in \mathbb{N}$, we define analogues of this notation for four-dimensional arrays: $\mathbb{A}^{k,\ell} \overset{\text{def}}{=} \mathbb{R}^{k \times k \times \ell \times \ell}$ denotes $k \times k \times \ell \times \ell$ arrays, $\mathbb{A}^{k,\ell}_{\geq 0}$ nonnegative arrays, and $\mathbb{A}^{k,\ell}_\Delta$ nonnegative arrays summing to 1; we also define 1- and ∞-norms for arrays. We typically use the letters A and B for four-dimensional arrays, and M and N for (two-dimensional) matrices.

2.2 (Directed) graphs, degrees, biases, and (directed) cuts

In this paper, we consider graphs without self-loops.[11] We give two slightly different definitions of graphs, which we call *unweighted* and *weighted* graphs. The algorithm we present handles unweighted graphs, but the analysis requires the more general notion of a weighted graph. We take care in lemma statements to indicate which kind of graph we are considering.

[11]This is because, from the perspective of Max-DICUT (which we are about to define), a self-loop edge is never satisfied by any assignment and is therefore uninteresting from an algorithmic perspective.

An *unweighted graph* $\mathcal{G}$ is given by a set $V(\mathcal{G})$ of vertices and a set $E(\mathcal{G}) \subseteq \{(u,v) : u \neq v \in V(\mathcal{G})\}$ of edges. We typically denote $n = |V|$ and $m = |E|$.

A *weighted graph* on a vertex-set $V = V(\mathcal{G})$ is defined by an *adjacency matrix* $\mathsf{AdjMat}_{\mathcal{G}} \in \mathbb{M}_{\geq 0}^{V \times V}$ with zeros on the diagonal. An unweighted graph $\mathcal{G}$ can be viewed as a weighted graph defined by an adjacency matrix $\mathsf{AdjMat}_{\mathcal{G}}$, where $\mathsf{AdjMat}_{\mathcal{G}}(u,v) = 1$ iff $(u,v) \in E(\mathcal{G})$. We let $m_{\mathcal{G}} = \sum_{u,v \in V} \mathsf{AdjMat}_{\mathcal{G}}(u,v)$ denote the total weight in a weighted graph $\mathcal{G}$.

Given a vertex $v \in V$ in a weighted graph $\mathcal{G}$, we define its *out-* and *in-degrees*

$$\mathsf{deg\text{-}out}_{\mathcal{G}}(v) \stackrel{\text{def}}{=} \sum_{u \in V} \mathsf{AdjMat}_{\mathcal{G}}(v,u) \text{ and } \mathsf{deg\text{-}in}_{\mathcal{G}}(v) \stackrel{\text{def}}{=} \sum_{u \in V} \mathsf{AdjMat}_{\mathcal{G}}(u,v),$$

and its *(total) degree*

$$\mathsf{deg}_{\mathcal{G}}(v) \stackrel{\text{def}}{=} \mathsf{deg\text{-}out}_{\mathcal{G}}(v) + \mathsf{deg\text{-}in}_{\mathcal{G}}(v).$$

If $\mathsf{deg}_{\mathcal{G}}(v) = 0$, we say v is *isolated*; otherwise, we define v's *bias*

$$\mathsf{bias}_{\mathcal{G}}(v) \stackrel{\text{def}}{=} \frac{\mathsf{deg\text{-}out}_{\mathcal{G}}(v) - \mathsf{deg\text{-}in}_{\mathcal{G}}(v)}{\mathsf{deg}_{\mathcal{G}}(v)} \in [-1,1].$$

Finally, for a "cut" $\mathbf{x} \in \{0,1\}^V$, we define its *value* in $\mathcal{G}$

$$\mathsf{val}_{\mathcal{G}}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{m_{\mathcal{G}}} \sum_{u,v \in V} \mathsf{AdjMat}_{\mathcal{G}}(u,v) x_v (1 - x_u),$$

and the overall $\mathsf{Max\text{-}DICUT}$ value of $\mathcal{G}$ as the maximum value of any cut:

$$\mathsf{val}_{\mathcal{G}}(\mathbf{x}) \stackrel{\text{def}}{=} \max_{\mathbf{x} \in \{0,1\}^V} \mathsf{val}_{\mathcal{G}}(\mathbf{x}).$$

2.3 Concentration

We write $\exp(x) = e^{-x}$. We shall need a number of concentration inequalities which operate in different parameter regimes of interest. We list several well-known inequalities as well as some convenient corollaries.

Lemma 2.1 (Chernoff upper bound)**.** *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables, and let $X = \sum_{i=1}^n X_i$. Then for all $\delta > 0$,*

$$\Pr[X \geq (1+\delta)\,\mathbb{E}[X]] \leq \exp(-\delta^2\,\mathbb{E}[X]/(2+\delta)).$$

Lemma 2.2 (Chernoff lower bound)**.** *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables, and let $X = \sum_{i=1}^n X_i$. Then for all $0 \leq \delta \leq 1$,*

$$\Pr[X \leq (1-\delta)\,\mathbb{E}[X]] \leq \exp(-\delta^2\,\mathbb{E}[X]/2).$$

Corollary 2.3 (Two-sided Chernoff bound)**.** *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables, and let $X = \sum_{i=1}^n X_i$. Then for all $0 \leq \delta \leq 1$,*

$$\Pr[|X - \mathbb{E}[X]| \geq \delta\,\mathbb{E}[X]] \leq 2\exp(-\delta^2\,\mathbb{E}[X]/3).$$

Corollary 2.4 (Chernoff upper bound, high deviation form)**.** *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables, and let $X = \sum_{i=1}^n X_i$. Then for all $\eta \geq 3\,\mathbb{E}[X]$,*

$$\Pr[X \geq \eta] \leq \exp(-\eta/8).$$

Lemma 2.5 (Weighted Chernoff bound [CL06, cf. Theorem 3.3])**.** *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables. Let $0 < \nu_1, \ldots, \nu_n$ be weights, and let $X = \sum_{i=1}^n \nu_i X_i$. Let $\lambda_0 = \max_i\{\nu_i\}$ and $\lambda_2 = \sum_{i=1}^n \nu_i^2 \mathbb{E}[X_i]$. Then for all $\delta > 0$,*

$$\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq \exp(-\delta^2 \mathbb{E}[X]^2/2\lambda_2)$$

and

$$\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq \exp(-\delta^2 \mathbb{E}[X]^2/(2\lambda_2 + \lambda_0\delta\,\mathbb{E}[X]).$$

Corollary 2.6 (Two-sided weighted Chernoff bound, low weights)**.** *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables. Let $0 < \nu_1, \ldots, \nu_n \leq 1$ be weights, and let $X = \sum_{i=1}^n \nu_i X_i$. Then for all $\delta > 0$,*

$$\Pr[|X - \mathbb{E}[X]| \geq \delta\,\mathbb{E}[X]] \leq 2\exp(-\delta^2 \mathbb{E}[X]/3).$$

Proof. Follows from the previous lemma since $\lambda_2 \leq \sum_{i=1}^n \nu_i \mathbb{E}[X_i] = \mathbb{E}[X]$ and $\lambda_0 \leq 1$. □

Lemma 2.7 (Chebyshev bound)**.** *Let $X_1, \ldots, X_n$ be random variables, and let $X = \sum_{i=1}^n X_i$. Then for all $\eta > 0$,*

$$\Pr[|X - \mathbb{E}[X]| \geq \eta] \leq \frac{\mathsf{Var}[X]}{\eta^2}.$$

Corollary 2.8 (Chebyshev with limited independence)**.** *Let $X_1, \ldots, X_n$ be random variables such that $0 \leq X_1, \ldots, X_n \leq 1$, and let $X = \sum_{i=1}^n X_i$. Further, suppose that each X_i is independent (pairwise) of all but D variables $\{X_j\}_{j\in[n]}$. Then for all $\eta > 0$,*

$$\Pr[|X - \mathbb{E}[X]| \geq \eta] \leq \frac{D \cdot \mathbb{E}[X]}{\eta^2}.$$

In particular, if the variables are pairwise independent, then

$$\Pr[|X - \mathbb{E}[X]| \geq \eta] \leq \frac{\mathbb{E}[X]}{\eta^2}.$$

Proof. Follows using $\mathsf{Var}[X] = \sum_{i,j=1}^n \mathbb{E}[X_i X_j] - \mathbb{E}[X_i]\mathbb{E}[X_j]$, the limited independence assumption, and the fact that for all $i, j \in [n]$, $\mathbb{E}[X_i X_j] \leq \mathbb{E}[X_i]$ (using $0 \leq X_i, X_j \leq 1$). □

2.4 Sparsification for Max-DICUT

The following lemma is a standard statement about sparsification for the Max-DICUT problem, which essentially lets us reduce to considering graphs with linearly many edges. We include the proof in Appendix A for completeness.

Lemma 2.9 (Linear sparsification preserves Max-DICUT values)**.** *There exists a universal constant $C_{\mathrm{spar}} > 0$ such that the following holds. For every $\epsilon_{\mathrm{spar}} \in (0,1)$ and $n, m \in \mathbb{N}$, suppose $C_{\mathrm{spar}} n/(\epsilon_{\mathrm{spar}}^2 m) \leq p_{\mathrm{spar}} \leq 1$. Then for every* unweighted *graph $\mathcal{G}$ on n vertices with m edges, if we let $\mathcal{G}_{\mathrm{spar}}$ be the random (unweighted) graph resulting from throwing away every edge of $\mathcal{G}$ independently with probability $1 - p_{\mathrm{spar}}$, then with probability $99/100$ over the choice of $\mathcal{G}_{\mathrm{spar}}$, we have $|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{G}_{\mathrm{spar}}}| \leq \epsilon_{\mathrm{spar}}$ and $|m_{\mathcal{G}_{\mathrm{spar}}} - p_{\mathrm{spar}} m| \leq \epsilon_{\mathrm{spar}} p_{\mathrm{spar}} m$.*

2.5 k-wise independent hash families

The following definition of a k-wise independent hash family will play a role in the algorithm we present in Section 4 below.

Definition 2.10. *A family of hash functions $\mathcal{H} = \{h : [n] \to [m]\}$ is* k-wise independent *if it satisfies the following properties:*

- *For every $x \in [n]$ and $a \in [m]$, and $h \sim \mathcal{H}(n,m)$ uniformly, $\Pr[h(x) = a] = \frac{1}{m}$, and*
- *For every distinct $x_1, \ldots, x_k \in [n]$, and $h \sim \mathcal{H}(n,m)$ uniformly, $h(x_1), \ldots, h(x_k)$ are independent random variables.*

Lemma 2.11 ([Jof74], see e.g. [SSSV23, §2.6])**.** *For every $k, n, m = 2^\ell \in \mathbb{N}$, there exists a family of k-wise independent hash functions $\mathcal{H}_k = \{h : [n] \to [m]\}$ such that a uniformly random hash function can be sampled with $O_k(\log n + \log m)$ bits of randomness.*

3 Reducing Max-DICUT approximation to "weak estimates" of the bias-degree array

In this section, we develop some machinery to reduce the Max-DICUT approximation problem for a graph $\mathcal{G}$ to a certain kind of "weak estimation" problem for a kind of "snapshot" of $\mathcal{G}$ called its "bias-degree array". In Section 3.2 below we outline this reduction, but to begin, we formally define snapshots and state the result of [FJ15] upon which we will rely.

3.1 Formally defining "snapshots"

3.1.1 Intervals and thresholds

Let $\mathbb{T}^\ell \subseteq \mathbb{R}^{\ell+1}$ denote the space of vectors $\mathbf{t} = (t_0, \ldots, t_\ell)$ such that $t_0 < \cdots < t_\ell$. We call such a vector a *threshold vector of length ℓ*. Given a threshold vector $\mathbf{t} \in \mathbb{T}^\ell$, for any $x \in [t_0, t_\ell]$, we define x's *index* $\mathsf{ind}^{\mathbf{t}}(x)$ (w.r.t. $\mathbf{t}$) as the unique $i \in [\ell]$ such that $t_{i-1} \le x < t_i$ (and if $x = t_\ell$ then $\mathsf{ind}^{\mathbf{t}}(x) = \ell$).

Suppose $\mathbf{t} = (t_0, \ldots, t_\ell) \in \mathbb{T}^\ell$ and $\mathbf{t}' = (t'_0, \ldots, t'_{\ell'}) \in \mathbb{T}^{\ell'}$ are two threshold vectors with $\ell' > \ell$. We say $\mathbf{t}'$ is a *refinement* of $\mathbf{t}$ if they have the same endpoints (i.e., $t_0 = t'_0$ and $t_\ell = t'_{\ell'}$) and for every $x, y \in [t_0, t_\ell]$, we have $\mathsf{ind}^{\mathbf{t}'}(x) = \mathsf{ind}^{\mathbf{t}'}(y) \implies \mathsf{ind}^{\mathbf{t}}(x) = \mathsf{ind}^{\mathbf{t}}(y)$.

We say $\mathbf{t}$ is *ϵ-wide* if for every $i \in [\ell]$, $\epsilon/2 \le t_i - t_{i-1} \le \epsilon$.

Fact 3.1. *For every $\ell \in \mathbb{N}$, $\mathbf{t} \in \mathbb{T}^\ell$, suppose $\epsilon = \max_{i \in [\ell]}(t_i - t_{i-1})$. Then for every $0 < \epsilon_{\text{bias}} \le \epsilon$, there exists some $\ell' \le \ell(1 + \epsilon/\epsilon_{\text{bias}}) \in \mathbb{N}$ and $\mathbf{t}' \in \mathbb{T}^\ell$ such that $\mathbf{t}'$ is a refinement of $\mathbf{t}$ and $\mathbf{t}'$ is ϵ_{bias}-wide.*

If along with a refinement $\mathbf{t}' \in \mathbb{T}^{\ell'}$ of $\mathbf{t} \in \mathbb{T}^\ell$ we are given some vector $\mathbf{r} = (r_1, \ldots, r_\ell) \in \mathbb{R}^\ell$, we can define an *inherited* vector $\mathbf{r}' = (r'_1, \ldots, r'_{\ell'}) \in \mathbb{R}^{\ell'}$ by $r'_i = r_{\mathsf{ind}_{t'_i}}$.

3.1.2 Bias matrices

Let $\mathbb{T}^\ell_{\pm 1} \subseteq \mathbb{T}^\ell$ denote the subset of threshold vectors with $t_0 = -1$ and $t_\ell = 1$. We think of such vectors as defining partitions of biases in graphs. For shorthand, given a weighted graph $\mathcal{G}$ and a (nonisolated) vertex v, we write $\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v) \overset{\text{def}}{=} \mathsf{ind}^{\mathbf{t}}(\mathsf{bias}_{\mathcal{G}}(v)) \in [\ell]$ for the index representing the "bias class" containing v, and given a pair of nonisolated vertices u, v, we write $\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v) \overset{\text{def}}{=} (\mathsf{ind}^{\mathbf{t}}(\mathsf{bias}_{\mathcal{G}}(u)), \mathsf{ind}^{\mathbf{t}}(\mathsf{bias}_{\mathcal{G}}(v))) \in [\ell]^2$ for their pair of bias classes.

Definition 3.2 (Bias matrix [SSSV23]). *Given a weighted graph $\mathcal{G}$ and threshold vector $\mathbf{t} \in \mathbb{T}^{\ell}_{\pm 1}$, we define the* bias matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}} \in \mathbb{M}^{\ell}_{\Delta}$ *by*

$$\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}(i,j) \overset{\text{def}}{=} \frac{1}{m_{\mathcal{G}}} \sum_{u,v=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v) \mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)=(i,j)}.$$ [12]

In other words, the matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ counts the fraction of edges in the graph between each pair of bias classes. Note that this is a *normalized* matrix, i.e., its entries sum to 1, unlike the adjacency matrix $\mathsf{AdjMat}_{\mathcal{G}}$. This is simply a matter of convenience.

The importance of the bias matrix is that it can be used to estimate the $\mathsf{Max\text{-}DICUT}$ value of a graph:

Lemma 3.3 (Feige and Jozeph [FJ15]). *There exists a constant $\alpha_{\mathrm{FJ}} \in (0.483, 0.4899)$, $\ell_{\mathrm{FJ}} \in \mathbb{N}$, a vector of bias thresholds $\mathbf{t}_{\mathrm{FJ}} \in \mathbb{T}^{\ell_{\mathrm{FJ}}}_{\pm 1}$, and a vector of probabilities $\mathbf{r}_{\mathrm{FJ}} = (r_1, \ldots, r_\ell) \in [0,1]^{\ell_{\mathrm{FJ}}}$ such that the following holds.*

For every weighted graph $\mathcal{G}$,

$$\alpha_{\mathrm{FJ}} \cdot \mathsf{val}_{\mathcal{G}} \le \sum_{i,j=1}^{\ell_{\mathrm{FJ}}} r_i(1-r_j)\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}(i,j) \le \mathsf{val}_{\mathcal{G}}.$$

Though we will not need this fact, we remark that the estimate for $\mathsf{val}_{\mathcal{G}}$ given in the lemma corresponds to a simple randomized assignment for $\mathcal{G}$, namely the so-called “oblivious” assignment which assigns each nonisolated vertex $v \in V(\mathcal{G})$ to 1 with probability r_i and 0 otherwise, where $i = \mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v)$ is its bias class.

We will rely on a “robust” version of the [FJ15] result which allows using an “estimate” for the bias matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ (in exchange for some loss in the approximation factor), and also allows using a “refinement” of the [FJ15] partition (as opposed to the partition itself). This takes the form of the following corollary, which is an analogue of [SSSV23, Corollary 3.5] except with 1-norm instead of ∞-norm errors. For completeness, we give a proof in Section 6.3.

Corollary 3.4 (“Robust [FJ15] with refinement”). *Let $\alpha_{\mathrm{FJ}}, \ell_{\mathrm{FJ}}, \mathbf{t}_{\mathrm{FJ}}, \mathbf{r}_{\mathrm{FJ}}$ be as in Lemma 3.3. Let $\ell \in \mathbb{N}$, let $\mathbf{t} \in \mathbb{T}^{\ell}$ be a refinement of $\mathbf{t}_{\mathrm{FJ}}$, and let $\mathbf{r} \in [0,1]^{\ell}$ be the vector inherited from $\mathbf{r}_{\mathrm{FJ}}$.*

Let $\mathcal{G}$ be a weighted graph on n vertices, and let $\widehat{M} \in \mathbb{M}^{\ell}$ be an estimate for $\mathcal{G}$'s bias matrix: $\|\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}} - \widehat{M}\|_1 \le \epsilon$. Then

$$\alpha_{\mathrm{FJ}} \cdot \mathsf{val}_{\mathcal{G}} - 2\epsilon \le \sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) - \epsilon \le \mathsf{val}_{\mathcal{G}}.$$

3.1.3 The bias-degree array

Next, we introduce a new and more refined version of a snapshot of a graph $\mathcal{G}$ which also takes into account the degrees of the vertices. Suppose we also have a threshold vector $\mathbf{d} \in \mathbb{T}^k$ partitions vertex degrees in $\mathcal{G}$, in the following sense: all nonisolated vertices in $\mathcal{G}$ have degree between d_0 and d_k. We define similar notations: For nonisolated v, we write $\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \overset{\text{def}}{=} \mathsf{ind}^{\mathbf{d}}(\mathsf{deg}_{\mathcal{G}}(v))$ for the “degree class” of v. (For notational convenience, if $\mathsf{deg}_{\mathcal{G}}(v) = 0$ we will write $\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) = -\infty$.) For nonisolated u, v, we define:

$$\mathsf{db\text{-}ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v) = (\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(u), \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v), \mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u), \mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v)). \tag{3.5}$$

[12] Note that $\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)$ is not defined if v or u is isolated. But in either case, $\mathsf{AdjMat}_{\mathcal{G}}(u,v)$ vanishes, so we adopt the convention of discarding these terms.

This lets us define:

Definition 3.6 (Bias-degree array)**.** *Given a weighted graph $\mathcal{G}$ and threshold vectors $\mathbf{t} \in \mathbb{T}^{\ell}_{\pm 1}$, $\mathbf{d} = (d_0, \ldots, d_k)$, such that every nonisolated vertex $v \in V(\mathcal{G})$ has $d_0 \le \mathsf{deg}_{\mathcal{G}}(v) \le d_k$, we define the* bias-degree array $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}} \in \mathbb{A}^{k,\ell}_{\Delta}$ *by*

$$\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}(a,b,i,j) \overset{\text{def}}{=} \frac{1}{m_{\mathcal{G}}} \sum_{u,v=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v) \mathbb{1}_{\mathsf{db\text{-}ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)=(a,b,i,j)}. \tag{3.7}$$

This array is only more informative than the bias matrix because the bias-matrix can be recovered via "projections". More precisely:

Definition 3.8 (Projecting arrays into matrices)**.** *Given an array $A \in \mathbb{A}^{k,\ell}$, we define a matrix* $\mathsf{Proj}(A) \in \mathbb{M}^{\ell}$ *by projecting onto the third and fourth coordinates, i.e.,*

$$(\mathsf{Proj}(A))(i,j) = \sum_{a,b=1}^{k} A(a,b,i,j).$$

Fact 3.9. *Let $\mathbf{d} = (d_0, \ldots, d_k) \in \mathbb{T}^k$ be a degree partition and let $\mathcal{G}$ be a weighted graph such that all nonisolated vertices have degree between d_0 and d_k. Then* $\mathsf{Proj}(\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}) = \mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$.

3.2 Outline

The algorithm of Feige and Jozeph [FJ15] (in the form of Corollary 3.4 above) reduces the task of estimating the Max-DICUT value of a graph $\mathcal{G}$ to estimating its bias matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ in 1-norm. Now, our goal is to further reduce this task to a notion which we call *weakly estimating the smoothing of the bias array* $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$.

The motivation for this reduction is that the final "weak estimation" task can be carried out directly in the streaming setting using an "edge-and-vertex-subsampling" strategy (as described in Section 1.5). We carry this out in the following two sections, which contain an explicit description of the algorithm (Section 4) and its analysis (Section 5). For now, we introduce some more definitions and lemmas and build towards a statement of a lemma, Lemma 3.22, which precisely states that this notion of "weak estimation" is sufficient. Proofs of this and several preliminary lemmas which we state in this section are postponed until Section 6 below.

The reduction from approximating the Max-DICUT value of a graph $\mathcal{G}$ proceeds as follows:

1. We reduce to estimating a "smoothed" version of $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ in 1-norm (see Lemma 3.13 below); each entry in the smoothed matrix is a (weighted) average over a "window" of surrounding entries in the original matrix.

2. We reduce this to estimating smoothed version of the *array* $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$ in 1-norm (see Proposition 3.16 below).

3. Finally, we reduce this to a certain notion of a "weak estimate", which roughly corresponds to allowing certain small mistakes in the estimate, in terms of which entries are counted and with what weights (see Corollary 3.21 below).

3.3 Defining windows

We begin with defining some notations for "windows" around entries in (1-dimensional) vectors, (2-dimensional) matrices, and (4-dimensional) arrays. These will correspond to indices to $[\ell]$, $[\ell]^2$, and $[k]^2 \times [\ell]^2$, respectively, where $k, \ell \in \mathbb{N}$. In each case, windows will correspond to a ball of a certain radius in the ∞-norm.

More concretely, we make the following definitions:

Definition 3.10 (Windows)**.** *Suppose $w < \ell \in \mathbb{N}$. For $i \in [\ell]$, let*

$$\mathsf{Win}^{w,\ell}(i) \overset{\text{def}}{=} \{i' \in [\ell] : |i' - i| \le w\}$$

denote the 1-dimensional window *around i. For $i, j \in [\ell]$, let*

$$\mathsf{Win}^{w,\ell}(i,j) \overset{\text{def}}{=} \mathsf{Win}^{w,\ell}(i) \times \mathsf{Win}^{w,\ell}(j) = \{(i',j') \in [\ell]^2 : \max\{|i'-i|, |j'-j|\} \le w\}$$

denote the 2-dimensional window *around (i,j). Given also $k > w \in \mathbb{N}$, for $a, b \in [k]$ and $i, j \in [\ell]$, let*

$$\begin{aligned}\mathsf{Win}^{w,k,\ell}(a,b,i,j) &\overset{\text{def}}{=} \mathsf{Win}^{w,k}(a,b) \times \mathsf{Win}^{w,\ell}(i,j) \\ &= \{(a',b',i',j') \in [k]^2 \times [\ell]^2 : \max\{|a-a'|, |b-b'|, |i-i'|, |j-j'|\} \le w\}\end{aligned}$$

denote the 4-dimensional window *around (a,b,i,j).*

We state various basic but useful facts about windows in Section 6.1 below.

3.4 Reducing approximating Max-DICUT value to estimating a "smoothed" bias matrix

We now define what it means to *smooth* a matrix $M \in \mathbb{M}^\ell$ over windows of size w.

Definition 3.11 (Smoothing matrices)**.** *Let $\ell \in \mathbb{N}$ and $M \in \mathbb{M}^\ell$. For $w < \ell \in \mathbb{N}$, we define a* smoothed *matrix $M^{\sim w} \in \mathbb{M}^\ell$ by*

$$M^{\sim w}(i,j) = \sum_{(i',j') \in \mathsf{Win}^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j') \cdot M(i',j'),$$

where $\nu^{\sim w,\ell}(i',j') \overset{\text{def}}{=} 1/|\mathsf{Win}^{w,\ell}(i',j')|$ is a normalization factor.

Note that the normalization factors $\nu^{\sim w,\ell}(i',j')$ do not depend on the matrix M. Informally, their importance is as follows. Suppose ℓ is even and $\ell > 2w$. Consider the entries $M(1,1)$ and $M(\ell/2, \ell/2)$. The former will contribute to $\approx w^2$ entries in $M^{\sim w}$ — in particular, the indices $\{1, \ldots, w+1\} \times \{1, \ldots, w+1\}$ — while the latter will contribute to $\approx 4w^2$ entries — in particular, $\{\ell/2 - (w+1), \ldots, \ell/2 + (w+1)\} \times \{\ell/2 - (w+1), \ldots, \ell/2 + (w+1)\}$. We will be interested in smoothing bias matrices, i.e., $M = \mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$, and we would like for the resulting matrices to "resemble" bias matrices, at least in the sense of still having entries summing to 1. In particular, the choice of normalization factors ensures that the following holds:

Proposition 3.12 (Smoothing preserves entry sum)**.** *For every $M \in \mathbb{M}^\ell$, $\sum_{i,j=1}^{\ell} M^{\sim w}(i,j) = \sum_{i,j=1}^{\ell} M(i,j)$. In particular, if $M \in \mathbb{M}^\ell_\Delta$ then $M^{\sim w} \in \mathbb{M}^\ell_\Delta$.*

The proof is by a simple double-counting argument; we give it in Section 6.2 below.

Next, the following key lemma roughly states that estimating the smoothed matrix $\mathsf{BiasMat}^{\sim w}_{\mathcal{G},\mathbf{t}}$ in 1-norm is sufficient for running the Feige-Jozeph algorithm [FJ15] (in the form of Corollary 3.4) on a weighted graph $\mathcal{G}$:

Lemma 3.13 ("Smoothing graphs"). *There exists a universal constant C_{smooth} such that the following holds. Let $\epsilon_{\text{bias}} > 0$, and $w < \ell \in \mathbb{N}$. Let $\mathbf{t} \in \mathbb{T}^\ell$ be ϵ_{bias}-wide.*

Let $\mathcal{G}$ be a weighted graph and $M = \mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ be its bias matrix. Then there exists a weighted graph $\mathcal{H}$ such that $|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{H}}| \le C_{\text{smooth}}\epsilon_{\text{bias}}(w+1)$ and the bias matrix $N = \mathsf{BiasMat}_{\mathcal{H},\mathbf{t}}$ satisfies $\|N - M^{\sim w}\|_1 \le C_{\text{smooth}}\epsilon_{\text{bias}}(w+1)$.

The proof of Lemma 3.13 is given in Section 6.5 below, and is somewhat involved: we essentially make many copies of each vertex in $\mathcal{G}$, and then "perturb" the bias of each copy by tweaking the weights of incident edges, in order to make the final bias matrix resemble $M^{\sim w}$.

Lemma 3.13 is useful because it implies *both* that (1) the "[FJ15] estimate" will be similar between $\mathcal{H}$'s actual bias matrix N and $\mathcal{G}$'s smoothed bias matrix $M^{\sim w}$, and that (2) the $\mathsf{Max\text{-}DICUT}$ values of $\mathcal{G}$ and $\mathcal{H}$ are similar. Since we also know (3) that the "[FJ15] estimate" for $\mathcal{H}$ approximates $\mathcal{H}$'s $\mathsf{Max\text{-}DICUT}$ value (by Corollary 3.4), these three facts together will imply that the "[FJ15] estimate" based on $M^{\sim w}$ approximates $\mathcal{G}$'s $\mathsf{Max\text{-}DICUT}$ value.

3.5 Reducing estimating the bias matrix to estimating the bias-degree array

Unfortunately, we will not be able to directly estimate entries of the smoothed bias matrix $\mathsf{BiasMat}^{\sim w}_{\mathcal{G},\mathbf{t}}$ using our sampling-based algorithm. Roughly, this is because there may be huge discrepancy between degrees of vertices (indeed, from $O(1)$ to $\Omega(n)$), and the subsampling parameters will depend on the degree. So, we reduce to the problem of estimating more refined quantities: smoothed versions of the *array* $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$. Our hope is that, if the degree partition is fine enough, these quantities can actually be estimated.

Definition 3.14 (Smoothing arrays). *Let $k, \ell \in \mathbb{N}$ and $A \in \mathbb{A}^{k,\ell}$. For $w < k, \ell \in \mathbb{N}$, we define a* smoothed *array $A^{\sim w} \in \mathbb{A}^{k,\ell}$ by*

$$A^{\sim w}(a,b,i,j) = \sum_{(a',b',i',j') \in \mathsf{Win}^{w,k,\ell}(a,b,i,j)} \nu^{\sim w,k,\ell}(a',b',i',j') \cdot A(a',b',i',j'),$$

where $\nu^{\sim w,k,\ell}(a',b',i',j') \stackrel{\text{def}}{=} 1/|\mathsf{Win}^{w,k,\ell}(a',b',i',j')|$ is a normalization factor.

We have also have an analogue of Proposition 3.12 (proof omitted for syntactic similarity):

Proposition 3.15. *For every $A \in \mathbb{A}^\ell$, $\sum_{a,b=1}^k \sum_{i,j=1}^\ell A^{\sim w}(a,b,i,j) = \sum_{a,b=1}^k \sum_{i,j=1}^\ell A(a,b,i,j)$. In particular, if $A \in \mathbb{A}^{k,\ell}_\Delta$ then $A^{\sim w} \in \mathbb{A}^{k,\ell}_\Delta$.*

The following definition and proposition roughly state that to estimate the matrix $\mathsf{BiasMat}^{\sim w}_{\mathcal{G},\mathbf{t}}$ in 1-norm, it suffices to instead estimate the array $\mathsf{BiasDegArr}^{\sim w}_{\mathcal{G},\mathbf{d},\mathbf{t}}$ in 1-norm.

Proposition 3.16. *For every $w < k, \ell \in \mathbb{N}$, and $A \in \mathbb{A}^{k,\ell}$, let $M = \mathsf{Proj}(A)$. Then $M^{\sim w} = \mathsf{Proj}(A^{\sim w})$. Moreover, for any array $\widehat{A} \in \mathbb{A}^{k,\ell}$, define $\widehat{M} = \mathsf{Proj}(\widehat{A})$. Then $\|\widehat{M} - M^{\sim w}\|_1 \le \|\widehat{A} - A^{\sim w}\|_1$.*

Together with Fact 3.9, this proposition does give the desired reduction from the problem of 1-norm estimation for $\mathsf{BiasMat}^{\sim w}_{\mathcal{G},\mathbf{t}}$ to 1-norm estimation for $\mathsf{BiasDegArr}^{\sim w}_{\mathcal{G},\mathbf{d},\mathbf{t}}$. We prove the proposition (again by double-counting) in Section 6.2 below.

3.6 A notion of "weakly estimating" the bias-degree array

Let $A \in \mathbb{A}^{k,\ell}_{\ge 0}$ be an array with nonnegative entries. For the final step in our reduction, we define a certain notion of a "weak estimate" for the smoothed array $A^{\sim w}$, and give a statement (Corollary 3.21 below) which roughly says that such a "weak" estimate suffices for a 1-norm estimate (with some loss in parameters). Such a "weak" estimate will be what we actually aim to achieve in the algorithm, with $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$.

The notion of "weak estimate" relies on the definition of two additional arrays $A^{-w}, A^{+w} \in \mathbb{A}^{k,\ell}_{\geq 0}$ which satisfy the inequality $A^{-w} \leq A^{\sim w} \leq A^{+w}$ entrywise and account for "off-by-one" errors when estimating $A^{\sim w}$. We first describe these arrays informally as the actual definitions may appear quite technical. Consider an "estimation" function $\xi : [k]^2 \times [\ell]^2 \to [k]^2 \times [\ell]^2$ for the graph $\mathcal{G}$, which takes as input the index (a, b, i, j) of an entry in the array A, and outputs a tuple $\xi(a, b, i, j)$ which is promised to differ from (a, b, i, j) by at most 1 in each entry. (This will correspond to issues with "borderline" vertices when $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$, as described in Section 1.5 above.) Now suppose we tried to estimate the entry $A^{\sim w}(a, b, i, j)$ using the expression

$$\sum_{\xi(a',b',i',j') \in \mathsf{Win}^{w,k,\ell}(a,b,i,j)} \nu^{\sim w,k,\ell}(a', b', i', j') \cdot A(a', b', i', j').$$

The quantities $A^{-w}(a, b, i, j)$ and $A^{+w}(a, b, i, j)$ are lower- and upper-bounds for this expression, respectively, based on the "worst possible" function ξ.

To be more precise, we define the arrays $A^{-w}, A^{+w} \in \mathbb{A}^{k,\ell}_{\geq 0}$ as follows:

Definition 3.17 (Upper- and lower-bound normalization factors)**.** *Let $w < k, \ell \in \mathbb{N}$. Then for $a', b' \in [k], i', j' \in [\ell]$, we define*

$$\nu^{-w,k,\ell}(a', b', i', j') \overset{\text{def}}{=} \min_{(a'',b'',i'',j'') \in \mathsf{Win}^{1,k,\ell}(a',b',i',j')} \nu^{\sim w,k,\ell}(a'', b'', i'', j''),$$

and

$$\nu^{+w,k,\ell}(a', b', i', j') \overset{\text{def}}{=} \max_{(a'',b'',i'',j'') \in \mathsf{Win}^{1,k,\ell}(a',b',i',j')} \nu^{\sim w,k,\ell}(a'', b'', i'', j'').$$

Definition 3.18 (Upper- and lower-bound arrays)**.** *Let $w < k, \ell \in \mathbb{N}$ and $A \in \mathbb{A}^{k,\ell}$. We define two arrays $A^{-w}, A^{+w} \in \mathbb{A}^{k,\ell}_{\geq 0}$ by defining, for all $a, b \in [k]$ and $i, j \in [\ell]$:*

$$A^{-w}(a, b, i, j) \overset{\text{def}}{=} \sum_{(a',b',i',j') \in \mathsf{Win}^{w-1,k,\ell}(a,b,i,j)} \nu^{-w,k,\ell}(a', b', i', j') A(a', b', i', j'),$$

and

$$A^{+w}(a, b, i, j) \overset{\text{def}}{=} \sum_{(a',b',i',j') \in \mathsf{Win}^{w+1,k,\ell}(a,b,i,j)} \nu^{+w,k,\ell}(a', b', i', j') A(a', b', i', j').$$

Note that by definition $\nu^{-w,k,\ell}(a', b', i', j') \leq \nu^{\sim w,k,\ell}(a', b', i', j') \leq \nu^{+w,k,\ell}(a', b', i', j')$ (since $(a', b', i', j') \in \mathsf{Win}^{1,k,\ell}(a', b', i', j')$). Hence, (since A has nonnegative entries) we have $A^{-w}(a, b, i, j) \leq A^{\sim w}(a, b, i, j) \leq A^{+w}(a, b, i, j)$, since they sum over windows around (a, b, i, j) of sizes $w - 1$, w, and $w + 1$, respectively, using increasing normalization factors. We also have the following simple lemma:

Lemma 3.19. *Let $w < k, \ell \in \mathbb{N}$ and $A \in \mathbb{A}^{k,\ell}_{\Delta}$. For all $a, b \in [k]$ and $i, j \in [\ell]$, we have*

$$A^{-w}(a, b, i, j) \leq A^{+w}(a, b, i, j) \leq 1.$$

Proof. For the first inequality, see above. For the second, note from Definitions 3.14 and 3.17 that $\nu^{+w,k,\ell}(a', b', i', j') \leq 1$ implying $A^{+w}(a, b, i, j) \leq \sum_{(a',b',i',j') \in \mathsf{Win}^{w+1,k,\ell}(a,b,i,j)} A(a', b', i', j') \leq 1$. $\square$

A second key lemma, which we prove in Section 6.4 below, is that we can control the error between these arrays:

Lemma 3.20 ("Sandwiching $A^{\sim w}$"). *There exists a universal constant $C_{\text{win}} > 0$ such that for every $w < k, \ell \in \mathbb{N}$ and every $A \in \mathbb{A}_{\Delta}^{k,\ell}$, $\|A^{+w} - A^{-w}\|_1 \leq C_{\text{win}}/w$.*

Immediately we get the following corollary, which states that a "sandwiched entrywise" estimate for $A^{\sim w}$ is also an estimate in 1-norm. Note that there is a substantial loss in parameters, corresponding to moving from ∞- to 1-norm.

Corollary 3.21. *Let $w < k, \ell \in \mathbb{N}$, $A \in \mathbb{A}_{\Delta}^{k,\ell}$, and $\epsilon > 0$. Suppose $\widehat{A} \in \mathbb{A}^{k,\ell}$ is such that*

$$A^{-w} - \epsilon \leq \widehat{A} \leq A^{+w} + \epsilon$$

entrywise. Then $\|\widehat{A} - A^{\sim w}\|_1 \leq \epsilon(k\ell)^2 + C_{\text{win}}/w$.

Proof. Since $A^{-w} - \epsilon \leq \widehat{A} \leq A^{+w} + \epsilon$ entrywise, we can pick some $\widehat{A}' \in \mathbb{A}^{k,\ell}$ such that $\|\widehat{A}' - \widehat{A}\|_\infty \leq \epsilon$ and $A^{-w} \leq \widehat{A}' \leq A^{+w}$ entrywise. The former implies $\|\widehat{A}' - \widehat{A}\|_1 \leq \epsilon(k\ell)^2$, while by Lemma 3.20 the latter implies $\|\widehat{A}' - A^{\sim w}\|_1 \leq C_{\text{win}}/w$. Hence, by the triangle inequality, $\|\widehat{A} - A^{\sim w}\|_1 \leq \epsilon + C_{\text{win}}/w$. □

3.7 Stating the reduction lemma (Lemma 3.22)

Finally, putting our work in the previous subsections together allows us to state our final "reduction lemma" as follows. The proof is essentially several triangle inequalities; we give it in Section 6.3.

Lemma 3.22 (Reducing $\mathsf{Max\text{-}DICUT}$ approximation for $\mathcal{G}$ to weak estimates of $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$). *For every $w < k, \ell \in \mathbb{N}$, $\epsilon_{\text{err}}, \epsilon_{\text{bias}} > 0$, the following holds. Let $\delta_{\text{bias}} = C_{\text{smooth}}\epsilon_{\text{bias}}(w+1)$ and $\delta_{\text{err}} = \epsilon_{\text{err}}(k\ell)^2 + C_{\text{win}}/w$, let $\delta = \delta_{\text{err}} + 2\delta_{\text{bias}}$, and let $\epsilon = 8\delta$. Let $\mathbf{d} = (d_0, \ldots, d_k) \in \mathbb{T}^k$. Let $\mathbf{t} = (t_0, \ldots, t_\ell) \in \mathbb{T}_{\pm 1}^{\ell}$ be a refinement of $\mathbf{t}_{\text{FJ}}$ which is ϵ_{bias}-wide. Let $\mathbf{r} = (r_1, \ldots, r_\ell) \in [0,1]^\ell$ be correspondingly inherited from $\mathbf{r}_{\text{FJ}} = (r_1, \ldots, r_{\ell_{\text{FJ}}})$.*

Let $\mathcal{G}$ be a weighted graph such that for every vertex $v \in V(\mathcal{G})$, either $\mathsf{deg}_{\mathcal{G}}(v) = 0$, or $d_0 \leq \mathsf{deg}_{\mathcal{G}}(v) \leq d_k$. Let $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$, and suppose $\widehat{A} \in \mathbb{A}^{k,\ell}$ is a "weak estimate" for $A^{\sim w}$, in the sense that

$$A^{-w} - \epsilon_{\text{err}} \leq \widehat{A} \leq A^{+w} + \epsilon_{\text{err}}$$

entrywise. Then for $\widehat{M} = \mathsf{Proj}(\widehat{A})$,

$$(\alpha_{\text{FJ}} - \epsilon)\mathsf{val}_{\mathcal{G}} \leq \sum_{i,j=1}^{\ell} r_i(1 - r_j)\widehat{M}(i,j) - \delta \leq \mathsf{val}_{\mathcal{G}}.$$

In the next section, we present an algorithm which uses the reduction in Lemma 3.22 to approximate the $\mathsf{Max\text{-}DICUT}$ value of an *unweighted* graph $\mathcal{G}$, up to a *fixed* constant approximation error ϵ. We briefly remark on the parameters with which Lemma 3.22 will be invoked. Since ϵ will a fixed constant, we will need to set $w = O(1/\epsilon)$ (in order for δ_{err} to be sufficiently small). Thus, we will need $\epsilon_{\text{bias}} = O(\epsilon^2)$ (to make δ_{bias} sufficiently small), forcing $\ell = O(1/\epsilon^2)$ (see Fact 3.1). Moreover, in order to get estimates in sufficiently small space, we will need a relatively fine degree partition $d_i/d_{i-1} = O(1)$ (we will use $d_i/d_{i-1} = 2$ for simplicity), forcing $k = \Theta(\log n)$. This means we will need to set $\epsilon_{\text{err}} = O(\epsilon^5/\log^2 n)$ in order for δ_{err} to be sufficiently small.

4 The algorithm

In this section, we present an algorithm that approximates the $\mathsf{Max\text{-}DICUT}$ value of an arbitrary unweighted graph, thereby proving Theorem 1.3 (see Lemma 4.2 below). The algorithm itself has to deal with a number of technical issues, so we begin by outlining the algorithm and giving a number of pointers which hopefully make it easier to digest.

4.1 Overview

Informal recap. The main concrete goal of the algorithm is to produce a "weak estimate" for the bias-degree array $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$ of a graph $\mathcal{G}$, in the sense of the hypotheses of Lemma 3.22 from the previous section. We quickly review the motivations behind this task from the discussion in Section 1.5.

At the highest level, we want to sample random edges in a graph $\mathcal{G}$ and estimate the biases of their endpoints, in order to estimate the bias matrix $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$; this is sufficient to approximate the $\mathsf{Max\text{-}DICUT}$ value of $\mathcal{G}$ using the algorithm of Feige and Jozeph [FJ15] (see Lemma 3.3). We aim to design a "sampling procedure" in which we sample several different sets $S_1, \ldots, S_k$ of vertices whose degree is "roughly" d_i for some partition $d_0 \leq \cdots \leq d_k$ of degrees in $\mathcal{G}$; we also store (sparse versions of) the neighborhoods of these vertices. Then, we aim to use the edges which showed up in these neighborhoods to estimate entries of the bias-degree array $\mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$. (However, we still have to deal with a number of errors corresponding to mis-estimating parameters of sampled edges, in particular the biases and degrees of their endpoints, and thus we can only get a "weak estimate".)

Outline of the algorithm. Via Lemma 3.22, we seek to estimate the entries of the *smoothed* bias-degree array $A^{\sim w}$ where $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$. Roughly, this measures how many edges $e = (u, v)$ are there in $\mathcal{G}$ such that $(\mathsf{ind}^{\mathbf{d}}(\mathsf{deg}_{\mathcal{G}}(u)), \mathsf{ind}^{\mathbf{d}}(\mathsf{deg}_{\mathcal{G}}(v)), \mathsf{ind}^{\mathbf{d}}(\mathsf{bias}_{\mathcal{G}}(u)), \mathsf{ind}^{\mathbf{d}}(\mathsf{bias}_{\mathcal{G}}(v)) \in \mathsf{Win}^{w,k,\ell}(a, b, i, j)$; that is, in $\mathcal{G}$, u's degree, v's degree, u's bias, and v's bias are within w intervals of a, b, i, j, respectively, with respect to some global partition $\mathbf{d}$ of degrees and $\mathbf{t}$ of biases. (Actually, we are summing normalization factors $\nu^{\sim w,\ell,k}$ over these edges, but this technical detail can be ignored for now.)

To do this, for each "layer" $a \in \{1, \ldots, k\}$, we fix some probabilities $q_a, p_a \in [0, 1]$. We subsample the edges of $\mathcal{G}$ with probability q_a and then subsample positive-degree vertices in $\mathcal{G}_a$ with probability p_a to get a subset of vertices, which we denoted $\texttt{vStored}_a$ in the algorithm. We hope to store the $\mathcal{G}_a$-neighborhoods of these vertices to get a subset of edges denoted $\texttt{eStored}_a$. Ideally, we can use $\texttt{eStored}_a$ to estimate the degrees and biases of vertices in $\texttt{vStored}_a$, and further, if we see an edge stored in $\texttt{eStored}_a$ between vertices in $\texttt{vStored}_a$ and $\texttt{vStored}_b$, we can count it in the appropriate entry of the bias-degree array. However, note that $\texttt{eStored}_a$ cannot necessarily store all neighbors for every vertex in $\texttt{vStored}_a$; in particular, while our edge subsampling ensures that $\texttt{vStored}_a$ is unlikely to contain vertices of $\mathcal{G}$-degree much less than d_a, it may still contain vertices of $\mathcal{G}$-degree much *greater* than d_a, in which case we can not hope to store all its neighbors.

Some pointers. It is very helpful to remember that there are two distinct sources of randomness in the algorithm: We "sparsify" the graph by subsampling edges, and then subsample which vertices we actually store. In the analysis we will first analyze this edge-subsampling, and then conditioned on certain "good outcomes" for the edge-subsampling we will analyze the vertex-subsampling. Correspondingly, it is useful to keep in mind the graph $\mathcal{G}_a$ consisting of all the edges sampled in layer a (each is sampled w.p. q_a) and the set $\mathcal{N}_a$ of positive-degree vertices in $\mathcal{G}_a$. (However, the actual streaming algorithm is not necessarily be able to store these sets as they can grow too large; it only stores subsets $\texttt{eStored}_a$ and $\texttt{vStored}_a$, respectively.)

When we actually see an edge $e = (u, v)$ in $\mathcal{G}_a$, how do we know what to do with it (i.e., which entry of the matrix should it contribute to)? Note that $\mathbb{E}[\mathsf{deg}_{\mathcal{G}_a}(v)] = q_a \mathsf{deg}_{\mathcal{G}}(v)$. Thus, we can hope to use $q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v)$ (which we call v's "apparent degree") as an estimate for $\mathsf{deg}_{\mathcal{G}}(v)$. Roughly, this should work out if $\mathsf{deg}_{\mathcal{G}}(v)$ is decently large; for instance, we can use the Chernoff bound to show that w.h.p. $\mathsf{deg}_{\mathcal{G}}(v)/2 \leq q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v) \leq 2\mathsf{deg}_{\mathcal{G}}(v)$, so the apparent degree moves by at most 1 interval relative to the actual degree. The same sort of analysis is necessary to analyze the bias of a vertex and ultimately prove that we get a so-called "weak estimate". This is asserted in Lemma 4.4 below.

Finally, we remark that the subsampling probabilities q_a and p_a decrease and increase, respectively, as a function of a. Indeed, for "typical" a, p_a will be inversely proportional to q_a, i.e., $p_a q_a$ will have a fixed value independent of a. However, since q_a and p_a are *probabilities* and are in particular at most 1, when a is small we will have to take $q_a = 1$ while when a is large we will take $p_a = 1$. That this is necessary is unsurprising if we consider "extremal" graphs, e.g. low-degree regular graphs (where we cannot hope to subsample edges without losing information) and graphs consisting of high-degree stars (where we cannot hope to subsample high-degree vertices without losing information).[13]

Some technicalities. We describe several more technical issues that occur in the algorithm, with an aim towards increasing readability. We encourage the reader to ignore all these details in the first pass over the algorithm and its analysis.

Firstly, we can essentially reduce to the case where the number of edges in $\mathcal{G}$ satisfies $\sqrt{n} \leq m \leq O(n)$; if m is smaller, we can simply store all edges in the graph using $\widetilde{O}(\sqrt{n})$ space, and if m is larger, we can globally sparsify while preserving the Max-DICUT value up to some small loss (see Lemma 2.9 above). Thus, our main subroutine (`DiCutEstimator` below) will operate under this kind of assumption, though we relax it in the final algorithm (`WrappedDiCutEstimator` below) for completeness.

Also, recall that a streaming algorithm in general is not given the number of edges in the stream as input. To solve this issue, as in [SSSV23], we rewrite the code of `DiCutEstimator` to use a value $\widehat{m}$ as an "estimate" for m when setting all parameters. The hope is that when $\widehat{m}$ roughly equals m (say, up to a factor of 2) the algorithm will still be correct; in particular, we will run the algorithm for several values of $\widehat{m}$ in parallel. If $\widehat{m}$ is too small we may end up storing too many edges for the space bound, while if $\widehat{m}$ is too large we may store too few edges for correctness. The latter is acceptable because after the stream finishes we do know the correct number of edges m and therefore can discard results for incorrect $\widehat{m}$. But to deal with the other case, we need to ensure that `DiCutEstimator` *never* uses too much space even if $\widehat{m}$ is much too small. Thus, we add a condition called "`overflow`" which is triggered when the set `eStored` grows too large (though we eventually have to show that this condition would not be triggered when $\widehat{m}$ is correct, see Lemma 5.3 below).

Finally, one remaining technical issue stems from the fact that when we want to subsample a set of nonisolated vertices in the subsampled graph $\mathcal{G}_a$, but we do not know the nonisolated vertices ahead of time. In particular, each time we see a *new* nonisolated vertex we want to toss a p-biased coin — but if we decide *not* to store a vertex, we need to "remember" this decision if we happen to see it again. This would be manageable if the algorithm had random access to the results of n biased coin flips, but this model would be somewhat nonstandard. Instead, as in [SSSV23], we observe that when proving concentration it is sufficient to have four-wise independence in the vertex-subsampling procedure, and thus, we decide whether to store a vertex by plugging it into a previously sampled four-wise independent hash function (see Lemma 2.11).

[13]This is independent of the issue of how to "weight" the contributions of extremal *entries* by normalization factors discussed in the previous section.

4.2 Describing the algorithm

The goal of this section is to prove Theorem 1.3. We begin by presenting an algorithm, called `DiCutEstimator`, for estimating the Max-DICUT value of a stream of edges corresponding to a graph $\mathcal{G}$ under certain "niceness" conditions regarding the number of edges in $\mathcal{G}$. Specifically, the algorithm `DiCutEstimator` depends on an "estimate" $\widehat{m}$ for the number of edges m in $\mathcal{G}$; for correctness, we require that $\widehat{m}$ is within a factor of roughly two of m, and for the space bound, we require that $\widehat{m}$ is $\Omega(\sqrt{n})$ and $O(n)$, where n is the number of vertices in $\mathcal{G}$. Reducing to this case is fairly standard — essentially a combination of guessing the correct $\widehat{m}$ and using sparsification — but for completeness we also present a full algorithm, called `WrappedDiCutEstimator`, which approximates the Max-DICUT value for arbitrary graphs. Theorem 1.3 follows immediately from the correctness and efficiency of this algorithm, which we state in Lemma 4.2. However, the correctness of `DiCutEstimator`, and by extension `WrappedDiCutEstimator`, will depend on a key correctness lemma (Lemma 4.4 below) whose proof is postponed until Section 5.

`DiCutEstimator` itself depends on two subroutines, called `LayeredSampler` and `BiasDegArrEstimator`, which we present next. The former is responsible for subsampling edges and vertices, and is the only subroutine that needs the stream $\boldsymbol{\sigma}$, while the latter is responsible for using the sampled information to estimate the bias-degree array. We also remark that `LayeredSampler` is the "space hog" of the algorithm: the final space usage will be determined by the number of edges and vertices stored by `LayeredSampler`, since the remaining code essentially just does arithmetic.[14] We also include several tables containing definitions of parameters to be used in the algorithms.

Notation	Value	Description
$\epsilon' > 0$	$\epsilon/(8(2 + C_{\text{win}} + 2C_{\text{smooth}}))$ where C_{win} and C_{smooth} are as in Lemmas 3.13 and 3.20	Scaled-down error (for use in triangle inequalities)
$w \in \mathbb{N}$	$1/\epsilon'$	Size of windows for smoothing
$\epsilon_{\text{bias}} > 0$	$\epsilon'/(w+1)$	Maximum width of intervals in the refinement $\mathbf{t}$ of $\mathbf{t}_{\text{FJ}}$ (see next line)
$\mathbf{t} \in \mathbb{T}^{\ell}$	Refinement of $\mathbf{t}_{\text{FJ}}$ into ℓ intervals of width at most ϵ_{bias}	See Fact 3.1
$\ell \in \mathbb{N}$	$\leq \ell_{\text{FJ}}(1 + 1/\epsilon_{\text{bias}})$	Number of intervals in $\mathbf{t}$, see Fact 3.1
$\mathbf{r} \in [-1,1]^{\ell}$	Assignment probabilities inherited from $\mathbf{r}_{\text{FJ}}$ w.r.t. refinement $\mathbf{t}$ of $\mathbf{t}_{\text{FJ}}$	

Table 1: Global parameters determined by ϵ alone.

[14] Actually, this is true for `DiCutEstimator`, but in `WrappedDiCutEstimator` there is an additional log factor from running $O(\log n)$ copies of `DiCutEstimator` in parallel.

Notation	Value	Description
$m_{\mathtt{min}} \in \mathbb{N}$	$\sqrt{n}$	Minimum number of edges handled in `DiCutEstimator`
$m_{\mathtt{max}} \in \mathbb{N}$	$C_{\text{spar}} n/(\epsilon')^2$ where C_{spar} is as in Lemma 2.9	Maximum number of edges handled in `DiCutEstimator` (see Lemma 2.9)
$k^* \in \mathbb{N}$	$6 \log\log n$	Number of degree intervals before we begin subsampling edges
$D \in \mathbb{N}$	2^{k^*+w+2}	W.h.p. bound on max-degree of "counted vertices" in subsampled graphs (parameter for space bound), note that $D = O_\epsilon(\log^6 n)$
`eCutoff`	$\log^7 n$	Maximum number of stored neighbors per vertex

Table 2: Global parameters determined by ϵ and n.

Notation	Value	Description
$k \in \mathbb{N}$	$\log(2\widehat{m})$	Number of degree intervals (we will have $\widehat{m} \le m_{\mathtt{max}}$ and thus, $k = O_\epsilon(\log n)$)
$\rho > 0$	$1000\sqrt{D} \cdot (k\ell)^3/\epsilon'$	Factor controlling space usage, note that $\rho = O_\epsilon(\log^6 n)$ (assuming $\widehat{m} \le m_{\mathtt{max}}$)
$p_0 > 0$	$\rho/\sqrt{\widehat{m}}$	Factor in vertex-subsampling probability
`vCutoff`	$10\rho\sqrt{2\widehat{m}}$	Maximum number of stored vertices per layer, note that `vCutoff` $= O_\epsilon(\sqrt{n}\log^6 n)$ (assuming $\widehat{m} \le m_{\mathtt{max}}$)

Table 3: Global parameters determined by ϵ, n, and $\widehat{m}$.

Algorithm 1 `DiCutEstimator`$(\boldsymbol{\sigma}; \epsilon, n, \widehat{m})$

Goal: Estimate the Max-DICUT value in a graph stream, given an estimate for the number of edges

```
1: Import parameters defined in Table 1.
2: Track the true number $m$ of edges in $\boldsymbol{\sigma}$.
3: Run LayeredSampler($\boldsymbol{\sigma}; \epsilon, n, \widehat{m}$). If it returns "overflow", output "overflow". Otherwise, let
   (eStored$_1$, vStored$_1$), ..., (eStored$_k$, vStored$_k$) denote its output.
4: Let $\widehat{A} \leftarrow$ BiasDegArrEstimator($m$, (eStored$_1$, vStored$_1$), ..., (eStored$_k$, vStored$_k$); $\epsilon, n, \widehat{m}$).
5: Let $\widehat{M} \leftarrow \mathsf{Proj}(\widehat{A}) \in \mathbb{M}^\ell$.    ▷ See Definition 3.8
6: Output $\widehat{v} \leftarrow \sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) - \epsilon'(1 + C_{\text{win}} + 2C_{\text{smooth}})$.    ▷ See Lemma 3.22
```

When the size of the graph is "nice" and correctly estimated by $\widehat{m}$, we claim that `DiCutEstimator` uses small space and (with high probability) outputs a good estimate of $\mathsf{val}_{\mathcal{G}}$:

Lemma 4.1 (Correctness and efficiency for `DiCutEstimator`)**.** *Let $\epsilon > 0$, $n, m, \widehat{m} \in \mathbb{N}$, and $\mathcal{G}$ be an unweighted graph on n vertices and m edges. Let $\boldsymbol{\sigma}$ be a stream consisting of edges of $\mathcal{G}$ in arbitrary order. If $m_{\mathtt{min}} \le \widehat{m} \le m_{\mathtt{max}}$, then running* `DiCutEstimator` *on the stream $\boldsymbol{\sigma}$ uses*

Notation	Value	Description
$d_a \in \mathbb{N}$	2^a	Degree partition. We also define $\mathbf{d} = (d_0, \ldots, d_k)$ where $d_0 = 1$.
$q_a \in [0, 1]$	$\min\{2^{k^*-a}, 1\}$	Edge-sampling probability
$p_a \in [0, 1]$	$\min\{p_0 q_a^{-1}, 1\}$	Vertex-sampling probability

Table 4: "Per-layer" parameters defined for all $a \in [k]$ and determined by ϵ, n, and $\widehat{m}$.

$\sqrt{n} \cdot \log^{O(1)}(n) \cdot 2^{O(1/\epsilon)}$ space. Moreover, if $m/2 \le \widehat{m} \le 2m$, then with probability at least $9/10$, the output $\widehat{v}$ satisfies $(\alpha_{\mathrm{FJ}} - 8(1 + C_{\mathrm{win}} + 2C_{\mathrm{smooth}})\epsilon')\mathsf{val}_{\mathcal{G}} \le \widehat{v} \le \mathsf{val}_{\mathcal{G}}$.

We will prove Lemma 4.1 in Section 4.3 below (modulo the proof of another lemma). Next, we present the "wrapper" algorithm which reduces the case of general input graphs to the case handled by `DiCutEstimator` (i.e., ensures the conditions in Lemma 4.1).

Algorithm 2 `WrappedDiCutEstimator`$(\boldsymbol{\sigma}; n, \epsilon)$

```
Goal: Estimate the Max-DICUT value in a general graph stream
 7: Import parameters defined in Tables 1 and 2.
 8: Track the true number $m$ of edges in $\boldsymbol{\sigma}$.
 9: In parallel, store the first $m_{\mathtt{min}}$ edges in $\boldsymbol{\sigma}$, and if $m \le m_{\mathtt{min}}$, compute the Max-DICUT value of
    the stored edges exactly and output this value.
10: for $t \in \{\log_{1.9} m_{\mathtt{min}}, \ldots, \log_{1.9} n^2\}$ in parallel do
11:     Define $\widehat{m} \leftarrow 1.9^t$                    ▷ Handle graphs with $\approx \widehat{m}$ edges
12:     Let $p_{\mathrm{spar}} \leftarrow \min\{m_{\mathtt{max}}/\widehat{m}, 1\}$
13:     Let $\boldsymbol{\sigma}_{\mathrm{spar}}$ subsample $\boldsymbol{\sigma}$ by including each edge independently w.p. $p_{\mathrm{spar}}$.
14:     Let $\widehat{m}_{\mathrm{spar}} \leftarrow \widehat{m} p_{\mathrm{spar}}$.              ▷ Note $\widehat{m}_{\mathrm{spar}} = \min\{\widehat{m}, m_{\mathtt{max}}\}$
15:     Run DiCutEstimator($\boldsymbol{\sigma}_{\mathrm{spar}}; \epsilon, n, \widehat{m}_{\mathrm{spar}}$), and if it does not return "overflow", let $\widehat{v}$ denote
    its output.
16:     If $\widehat{m} \le m \le 1.9\widehat{m}$, output $\widehat{v} - \epsilon'$.
17: end for
```

The desired properties of `WrappedDiCutEstimator` are asserted in the following lemma:

Lemma 4.2 (Correctness and efficiency for `WrappedDiCutEstimator`)**.** *Let $\epsilon > 0$, $n \in \mathbb{N}$, and $\mathcal{G}$ be an unweighted graph on n vertices. Let $\boldsymbol{\sigma}$ be a stream consisting of edges of $\mathcal{G}$ in arbitrary order. Then running `WrappedDiCutEstimator` on the stream $\boldsymbol{\sigma}$ uses $\sqrt{n} \cdot \log^{O(1)}(n) \cdot 2^{O(1/\epsilon)}$ space. Moreover, with probability at least $9/10$, its output $\widehat{v}$ satisfies $(\alpha_{\mathrm{FJ}} - \epsilon)\mathsf{val}_{\mathcal{G}} \le \widehat{v} \le \mathsf{val}_{\mathcal{G}}$.*

We also prove Lemma 4.2 in Section 4.3 below, which completes the proof of our main theorem.

Now, we delve into the "meat" of the algorithm, beginning with the sampling algorithm `LayeredSampler`, which, roughly speaking, subsamples vertices and edges for us to store.

Algorithm 3 $\texttt{LayeredSampler}(\boldsymbol{\sigma}; \epsilon, n, \widehat{m})$

```
Goal: Output a “representative sample” of vertices with various degrees and their neighborhoods
18: Import parameters defined in Tables 1 to 4.
19: For all $a \in [k]$, initialize $\texttt{eStored}_a, \texttt{vStored}_a \leftarrow \emptyset$ and sample a hash function $\pi_a : [n] \to [1/p_a]$
    from $\mathcal{H}_4(n, 1/p_a)$ (see Lemma 2.11).[15]
20: Keep track of the number of edges $m$ in $\boldsymbol{\sigma}$.
21: for each edge $e = (u, v) \in \boldsymbol{\sigma}$ do
22:     for $a = 1, \ldots, k$ do
23:         Toss a biased coin which is 1 with probability $q_a$, and let $z$ denote its output.
24:         if $z = 1$ then
25:             for $v' \in \{u, v\}$ do
26:                 If $\pi_a(v') = 1$, set $\texttt{vStored}_a \leftarrow \texttt{vStored}_a \cup \{v'\}$.
27:                 If $|\texttt{vStored}_a| > \texttt{vCutoff}$, halt subroutine and return “overflow”.
28:                 If $v' \in \texttt{vStored}_a$ and $\texttt{deg}_{\texttt{eStored}_a}(v') < \texttt{eCutoff}$, set $\texttt{eStored}_a \leftarrow \texttt{eStored}_a \cup \{e\}$.
29:             end for
30:         end if
31:     end for
32: end for
Output: $m, (\texttt{eStored}_1, \texttt{vStored}_1), \ldots, (\texttt{eStored}_k, \texttt{vStored}_k)$.
```

As mentioned above, the space usage of LayeredSampler essentially determines the space usage of the entire algorithm, and this space usage is small because of the bound on the number of stored vertices and edges in Lines 27 and 28. Finally, here is the algorithm BiasDegArrEstimator which uses the output of LayeredSampler to estimate the (smoothed) bias-degree array.

[15]We assume for simplicity that $1/p_a$ is a power of 2. This could be enforced formally via a more careful choice of $\widehat{m}$ and D.

Algorithm 4 $\texttt{BiasDegArrEstimator}(m, (\texttt{eStored}_1, \texttt{vStored}_1), \ldots, (\texttt{eStored}_k, \texttt{vStored}_k); \epsilon, n, \widehat{m})$

```
33: Import parameters defined in Tables 1, 3 and 4.
34: For all $a \in [k]$ and $v \in \texttt{vStored}_a$, define:    ▷ Apparent $\mathcal{G}$-degree and $\mathcal{G}$-bias of $v$ respectively.
```

$$\texttt{dEst}_a(v) = \min\{q_a^{-1} \cdot \mathsf{deg}_{\texttt{eStored}_a}(v), d_k\} \quad \text{and} \quad \texttt{bEst}_a(v) = \mathsf{bias}_{\texttt{eStored}_a}(v).$$

```
35: For all $a \in [k]$ and $i \in [\ell]$, define:
```

$$\texttt{vEst}_{a,i} = \Big\{v \in \texttt{vStored}_a \mid \mathsf{deg}_{\texttt{eStored}_a}(v) < \texttt{eCutoff} \\ \wedge \mathsf{ind}^{\mathbf{d}}(\texttt{dEst}_a(v)) \in \mathsf{Win}^{w,k}(a) \wedge \mathsf{ind}^{\mathbf{t}}(\texttt{bEst}_a(v)) \in \mathsf{Win}^{w,\ell}(i)\Big\}.$$

```
36: For all $a, b \in [k]$ and $i, j \in [\ell]$, define:
```

$$\texttt{AEst}_{a,b,i,j} = \sum_{(u,v)\in \texttt{eEst}_{a,b,i,j}} \nu\texttt{Est}_{a,b}^{w,k,l}(u,v),$$

where:

$$\texttt{eEst}_{a,b,i,j} = \texttt{eStored}_{\min\{a,b\}} \cap (\texttt{vEst}_{a,i} \times \texttt{vEst}_{b,j}),$$

and

$$\nu\texttt{Est}_{a,b}^{w,k,l}(u,v) = \nu^{\sim w,k,\ell}\Big(\mathsf{ind}^{\mathbf{d}}(\texttt{dEst}_a(u)), \mathsf{ind}^{\mathbf{d}}(\texttt{dEst}_b(v)), \mathsf{ind}^{\mathbf{t}}(\texttt{bEst}_a(u)), \mathsf{ind}^{\mathbf{t}}(\texttt{bEst}_b(v))\Big).$$

Output: The array $\widehat{A} \in \mathbb{A}_{\geq 0}^{k,\ell}$ defined by

$$\widehat{A}(a,b,i,j) = \frac{\texttt{AEst}_{a,b,i,j}}{m q_{\min\{a,b\}} p_a p_b}. \tag{4.3}$$

The key claim about $\texttt{BiasDegArrEstimator}$ is that its output is a good estimate for $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}^{\sim w}$:

Lemma 4.4. *Let $\epsilon > 0, n, m, \widehat{m} \in \mathbb{N}$, and suppose that $m/2 \leq \widehat{m} \leq 2m$. Let $\mathcal{G}$ be an unweighted graph on n vertices with m edges and let $\boldsymbol{\sigma}$ be a stream consisting of edges of $\mathcal{G}$ in arbitrary order. Then, with probability at least $99/100$, for all $a, b \in [k]$ and $i, j \in [\ell]$, it holds that:*

$$A^{-w}(a,b,i,j) - \epsilon'/(k\ell)^2 \leq \widehat{A}(a,b,i,j) \leq A^{+w}(a,b,i,j) + \epsilon'/(k\ell)^2,$$

where $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$ and $\widehat{A} = \texttt{BiasDegArrEstimator}(\texttt{LayeredSampler}(\boldsymbol{\sigma}; \epsilon, n, \widehat{m}); \epsilon, n, \widehat{m})$.

The proof of this lemma is more involved, and is postponed until Section 5. Assuming Lemma 4.4 for now, we finish the proofs of Lemmas 4.1 and 4.2.

4.3 Proofs of Lemmas 4.1 and 4.2

Proof of Lemma 4.1 assuming Lemma 4.4. First, we prove the space bound. Indeed, $\texttt{LayeredSampler}$ outputs "$\texttt{overflow}$" immediately whenever any $\texttt{vStored}_a$'s size exceeds $\texttt{vCutoff} = 10\rho\sqrt{2\widehat{m}}$. Since we assumed $\widehat{m} \leq m_{\texttt{max}} = O(n/\epsilon^2)$, and defined $\rho \leq \log^7 n \cdot 2^{O(1/\epsilon)}$, we conclude that $|\texttt{vStored}_a| = \sqrt{n} \cdot \log^{O(1)} n \cdot 2^{O(1/\epsilon)}$. Moreover, $\texttt{LayeredSampler}$ stores at most $\texttt{eStored} = O(\log^7 n)$ neighbors of each vertex in $\texttt{vStored}_a$, uses $O(\log n)$ bits per neighbor, and finally, there are $k = O_\epsilon(\log n)$ values of a. This proves the space bound.

To prove correctness, we can now assume $m/2 \le \widehat{m} \le 2m$. We apply Lemma 4.4 to conclude that the output $\widehat{A}$ of `BiasDegArrEstimator` lies between A^{-w} and A^{+w} entrywise up to additive error $\epsilon_{\text{err}} = \epsilon'/(k\ell)^2$ where $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$. Then, we can apply Lemma 3.22, and we conclude that for $\widehat{M} = \mathsf{Proj}(\widehat{A})$, we have:

$$(\alpha_{\text{FJ}} - 8(1 + C_{\text{win}} + 2C_{\text{smooth}})\epsilon')\mathsf{val}_{\mathcal{G}} \le \sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) - \delta \le \mathsf{val}_{\mathcal{G}}.$$

□

Proof of Lemma 4.2 assuming Lemma 4.4. First, we prove the space bound. On Line 9 we only remember $m_{\mathtt{min}} = \sqrt{n}$ edges. Further, there are only $O(\log n)$ values of t. For each such value, we run `DiCutEstimator` with $m_{\mathtt{min}} \le \widehat{m}_{\text{spar}} = \min\{\widehat{m}, m_{\mathtt{max}}\} \le m_{\mathtt{max}}$. Thus, we get the desired space bound for `DiCutEstimator` using Lemma 4.1. Putting these together gives the desired bound for `WrappedDiCutEstimator`.

Now, we prove correctness. Note that if $m \le m_{\mathtt{min}}$, our algorithm always computes the $\mathsf{Max\text{-}DICUT}$ value exactly. Otherwise, since $\widehat{m}$ steps in powers of 1.9, there is a *unique* $\widehat{m}$ such that

$$\widehat{m} \le m \le 1.9\widehat{m}. \tag{4.5}$$

Since all other values of $\widehat{m}$ are ignored by Line 16, it suffices to consider only this value of $\widehat{m}$ and show that we output a correct estimate. To do so, we use Lemmas 2.9 and 4.1.

In particular, let $\mathcal{G}_{\text{spar}}$ denote the graph corresponding to $\boldsymbol{\sigma}_{\text{spar}}$, so that $\mathcal{G}_{\text{spar}}$ is formed from $\mathcal{G}$ by sampling each edge independently w.p. p_{spar}. Let $m_{\text{spar}} = m_{\mathcal{G}_{\text{spar}}}$. We claim that with probability 999/100 over the choice of $\mathcal{G}_{\text{spar}}$,

$$|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{G}_{\text{spar}}}| \le \epsilon' \tag{4.6}$$

and

$$|m_{\text{spar}} - p_{\text{spar}} m| \le \epsilon' p_{\text{spar}} m. \tag{4.7}$$

Indeed, if $p_{\text{spar}} = 1$ then $\mathcal{G}_{\text{spar}} = \mathcal{G}$ so the statement holds trivially. Otherwise, $p_{\text{spar}} = m_{\mathtt{max}}/\widehat{m}$ so that $\widehat{m}_{\text{spar}} = m_{\mathtt{max}}$. Now we can apply Lemma 2.9 with $\epsilon_{\text{spar}} = \epsilon'$, since

$$p_{\text{spar}} = \frac{m_{\mathtt{max}}}{\widehat{m}} \ge \frac{C_{\text{spar}} n}{(\epsilon')^2 m}$$

using the definition of $m_{\mathtt{max}}$ and the assumption $m \ge \widehat{m}$ (Eq. (4.5)).

Finally, we condition on $\mathcal{G}_{\text{spar}}$ such that Eqs. (4.6) and (4.7) occur, and we want to apply Lemma 4.1 to $\mathcal{G}_{\text{spar}}$. To do this, we have to check that

$$m_{\mathtt{min}} \le \widehat{m}_{\text{spar}} \le m_{\mathtt{max}} \tag{4.8}$$

and

$$m_{\text{spar}}/2 \le \widehat{m}_{\text{spar}} \le 2m_{\text{spar}}. \tag{4.9}$$

Indeed, Eq. (4.8) follows from the fact that $\widehat{m}_{\text{spar}} = \min\{\widehat{m}, m_{\mathtt{max}}\}$ and $\widehat{m} \ge m_{\mathtt{min}}$ (by definition of $\widehat{m}$ on Line 10). For Eq. (4.9), we multiply Eq. (4.5) by p_{spar} and add Eq. (4.7) to get

$$\widehat{m}_{\text{spar}} - \epsilon' p_{\text{spar}} m \le m_{\text{spar}} \le 1.9\widehat{m}_{\text{spar}} + \epsilon' p_{\text{spar}} m$$

which is sufficient because by Eq. (4.5), $\epsilon' p_{\text{spar}} m \le 1.9\epsilon' \widehat{m}_{\text{spar}} < .1\widehat{m}_{\text{spar}}$ since $\epsilon' < 0.01$ by definition.

Thus, by Lemma 4.1, the output $\widehat{v}$ of `DiCutEstimator` on $\mathcal{G}_{\text{spar}}$ satisfies $(\alpha_{\text{FJ}} - 8\epsilon'(1 + C_{\text{win}} + 2C_{\text{smooth}}))\mathsf{val}_{\mathcal{G}_{\text{spar}}} \le \widehat{v} \le \mathsf{val}_{\mathcal{G}_{\text{spar}}}$. We can add this inequality with Eq. (4.6) to conclude that

$$(\alpha_{\text{FJ}} - \epsilon)\mathsf{val}_{\mathcal{G}} = (\alpha_{\text{FJ}} - 8\epsilon'(2 + C_{\text{win}} + 2C_{\text{smooth}}))\mathsf{val}_{\mathcal{G}} \le \widehat{v} - \epsilon' \le \mathsf{val}_{\mathcal{G}},$$

as desired. □

5 Correctness of `BiasDegArrEstimator`: Proving Lemma 4.4

The goal of this section is to prove Lemma 4.4, which says that the estimated array $\widehat{A}$ lies between the arrays $\mathsf{BiasDegArr}^{-w}_{\mathcal{G},\mathbf{d},\mathbf{t}}$ and $\mathsf{BiasDegArr}^{+w}_{\mathcal{G},\mathbf{d},\mathbf{t}}$, up to some additive error. Throughout this section, we work with a fixed $\epsilon, n, \widehat{m}, \boldsymbol{\sigma}$ which also fixes $\mathcal{G}$ and m. Besides the variables in Tables 1 to 4, we also define the following notation: For $a \in [k]$, $\mathcal{G}_a$ will denote the random variable denoting the subgraph of $\mathcal{G}$ containing all the edges for which $z = 1$ in Line 23 in the a-th execution of the loop. We will use m_a to denote the random variable equal to the number of edges in $\mathcal{G}_a$ and $\mathcal{N}_a$ to denote the set-valued random variable that is equal to the set of all non-isolated vertices in $\mathcal{G}_a$. Note that these random variables are all determined by the randomness in Line 23.

For $a \in [k]$, we also define the set-valued random variable $S_a = \{v \in [n] : \pi_a(v) = 1\}$, that is, S_a is the set of all vertices that hash to 1 under π_a. Note that S_a is determined by the randomness in Line 19. Also, note that the randomness' in Lines 19 and 23 are independent.

The plan of attack is to build up to the full proof by defining several "good" events, that each happen with high probability and together imply that we get the desired output. We start by analyzing the condition in which our algorithm returns `overflow` in Section 5.1. In Sections 5.2 and 5.3, we analyze the degrees and the biases (respectively) of the vertices the $\mathcal{G}_a$ and show that they are close to the corresponding values in $\mathcal{G}$. By Lemma 5.16, this implies that our estimates ν`Est` in Line 36 are in the desired range. Finally, in Sections 5.4 to 5.6, we build on these lemmas and show that our final output is as desired.

5.1 Overflow condition in Algorithm 3

The goal of this section is to define and analyze two events $\mathcal{E}_{\texttt{of1}}$ and $\mathcal{E}_{\texttt{of2}}$ such that Algorithm 3 never returns `overflow` when they occur. These are defined in Lemma 5.1 and Lemma 5.2 respectively.

Lemma 5.1. *We have* $\Pr(\mathcal{E}_{\texttt{of1}}) \geq 999/1000$*, where the probability is over the randomness in Line 23 and* $\mathcal{E}_{\texttt{of1}}$ *is defined as follows: For every* $a \in [k]$*, we have:*

$$m_a \leq \begin{cases} 2q_a m, & \text{if } p_0 \leq q_a \\ 5\rho\sqrt{m}, & \text{if } p_0 > q_a \end{cases}.$$

Proof. Note that for all $a \in [k]$, the random variable m_a is the sum of m independent and identically distributed indicator random variables that take the value 1 with probability q_a. For all $a \in [k]$ such that $p_0 \leq q_a$, this fact together with Lemma 2.1 gives:

$$\Pr(m_a \geq 2q_a m) \leq \exp(-q_a m/3) \leq \exp(-\rho\sqrt{m}/3) \leq o(1/k).$$

Similarly, for all $a \in [k]$ such that $p_0 > q_a$ which implies $5\rho\sqrt{m} \geq 3p_0 m \geq 3q_a m$, Corollary 2.4 gives:

$$\Pr\bigl(m_a \geq 5\rho\sqrt{m}\bigr) \leq \exp\bigl(-5\rho\sqrt{m}/8\bigr) \leq o(1/k).$$

The lemma now follows by combining the two inequalities above and applying a union bound over all $a \in [k]$. □

Next, let $\mathcal{G}_1, \ldots, \mathcal{G}_k$ be a set of subgraphs of $\mathcal{G}$ that may be sampled in Line 23. For convenience, we shall often abbreviate $\mathcal{G}_{[k]} = \mathcal{G}_1, \ldots, \mathcal{G}_k$. Observe that $\mathcal{G}_{[k]}$ determines whether or not $\mathcal{E}_{\texttt{of1}}$ occurs. We have:

Lemma 5.2. *Fix any $\mathcal{G}_{[k]}$ such that $\mathcal{E}_{\texttt{of1}}$ occurs. We have $\Pr\left(\mathcal{E}_{\texttt{of2}}^{\mathcal{G}_{[k]}}\right) \geq 999/1000$, where the probability is over the randomness in Line 19 and $\mathcal{E}_{\texttt{of2}}^{\mathcal{G}_{[k]}}$ is defined as follows: For every $a \in [k]$, we have:*

$$|\mathcal{N}_a \cap S_a| < \texttt{vCutoff}.$$

Proof. Fix $\mathcal{G}_{[k]}$ as in the lemma. We show the lemma by showing that for all $a \in [k]$, we have $\Pr(|\mathcal{N}_a \cap S_a| \geq \texttt{vCutoff}) \leq o(1/k)$. Indeed, the lemma then follows by a union bound. Fix $a \in [k]$. If $p_0 > q_a$, we have by Lemma 5.1 that $|\mathcal{N}_a| \leq 2m_a \leq 10\rho\sqrt{m} < \texttt{vCutoff}$ and the result follows. Thus, assume that $p_0 \leq q_a$. As $\mathcal{E}_{\texttt{of1}}$ occurs, we have $|\mathcal{N}_a| \leq 2m_a \leq 4q_a m$ by Lemma 5.1. For all $v \in \mathcal{N}_a$, define the indicator random variable I_v to be 1 if and only if $v \in S_a$. Note that these random variables are pairwise independent as Line 19 samples a 4-wise independent hash function. Define $I = \sum_{v \in \mathcal{N}_a} I_v$ and note that (as I is a sum of pairwise independent indicator random variables) $\mathsf{Var}[I] \leq \mathbb{E}[I] \leq 4p_a q_a m = 4p_0 m$ as $p_0 \leq q_a$. We get:

$$\begin{aligned}
\Pr(|\mathcal{N}_a \cap S_a| \geq \texttt{vCutoff}) &= \Pr(I \geq \texttt{vCutoff}) \\
&\leq \Pr(I \geq 5p_0 m) && (\text{As } \texttt{vCutoff} \geq 10\rho\sqrt{\widehat{m}} = 10p_0\widehat{m} \geq 5p_0 m) \\
&\leq \frac{\mathsf{Var}[I]}{(p_0 m)^2} && (\text{Lemma 2.7 and } \mathbb{E}[I] \leq 4p_0 m) \\
&\leq \frac{4}{p_0 m} && (\text{As } \mathsf{Var}[I] \leq 4p_0 m) \\
&\leq \frac{8}{\rho\sqrt{m}} = o(1/k).
\end{aligned}$$

□

We now formalize our claim above that Algorithm 3 never returns $\texttt{overflow}$ if $\mathcal{E}_{\texttt{of1}}$ and $\mathcal{E}_{\texttt{of2}}$ occur.

Lemma 5.3. *Fix any $\mathcal{G}_{[k]}$ such that $\mathcal{E}_{\texttt{of1}}$ occurs. If $\mathcal{G}_{[k]}$ is sampled in Line 23 and $\mathcal{E}_{\texttt{of2}}^{\mathcal{G}_{[k]}}$ occurs, then the following hold (with probability 1):*

1. *Algorithm 3 does not return* $\texttt{overflow}$.
2. *For all $a \in [k]$, we have $\texttt{vStored}_a = \mathcal{N}_a \cap S_a$.*
3. *For all $a \in [k]$ and $u \in \texttt{vStored}_a$ such that $\mathsf{deg}_{\texttt{eStored}_a}(u) < \texttt{eCutoff}$, we have for all $v \in [n]$ that $(u, v) \in \texttt{eStored}_a \iff (u, v) \in \mathcal{G}_a$ and $(v, u) \in \texttt{eStored}_a \iff (v, u) \in \mathcal{G}_a$.*
4. *For all $a \in [k]$, $u \in \texttt{vStored}_a$, we have $\mathsf{deg}_{\texttt{eStored}_a}(u) < \texttt{eCutoff} \iff \mathsf{deg}_{\mathcal{G}_a}(u) < \texttt{eCutoff}$.*

Proof. We fix the randomness in Lines 19 and 23 arbitrarily such that both $\mathcal{G}_{[k]}$ and $\mathcal{E}_{\texttt{of2}}^{\mathcal{G}_{[k]}}$ occur and show that Items 1 to 4 hold. Note that fixing the randomness fixes the value of all variables in Algorithms 3 and 4. We show each part in turn.

1. Note that Algorithm 3 returns $\texttt{overflow}$ only if there exists $a \in [k]$ such that at least $\texttt{vCutoff}$ non-isolated vertices v in $\mathcal{G}_a$ satisfy $v \in S_a$, or equivalently, if $|\mathcal{N}_a \cap S_a| \geq \texttt{vCutoff}$. By Lemma 5.2, this can never happen if $\mathcal{E}_{\texttt{of2}}^{\mathcal{G}_{[k]}}$ occurs.
2. Follows directly from Item 1.

3. As the $\Longrightarrow$ direction is trivial, we only show the $\Longleftarrow$ direction. We only show the first equivalence as the proof for the second one is analogous. Let $v \in [n]$ be arbitrary such that $(u, v) \in \mathcal{G}_a$. Due to Item 1, we are guaranteed that the algorithm executes Line 28 with $e = (u, v)$ and a and $v' = u$. The "if" in Line 28 is satisfied due to Line 26 and the fact that $u \in \texttt{vStored}_a$ and we get $(u, v) \in \texttt{eStored}_a$, as desired.

4. Follows directly from Item 3.

$\square$

5.2 Degrees after edge-subsampling

For $a \in [k]$ and $v \in [n]$, define a random variable for the *apparent degree index* as follows:

$$\widehat{\mathsf{d}\text{-}\mathsf{ind}}^{\mathsf{d}}_a(v) = \mathsf{ind}^{\mathsf{d}}\big(\min\{q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v), d_k\}\big). \tag{5.4}$$

In the above definition, we adopt the convention that the right-hand side is $-\infty$ if $\mathsf{deg}_{\mathcal{G}_a}(v) = 0$. We remark that this definition is similar to but different from the quantity $\texttt{dEst}_a(v)$ we defined in Algorithm 4, since that was only defined for vertices in $\texttt{vStored}_a$ and counted only the edges in $\texttt{eStored}_a$. We begin with the following lemma, which gives a general characterization of degrees of vertices in the graphs $\mathcal{G}_a$.

Lemma 5.5. *We have $\Pr\big(\mathcal{E}_{\mathtt{deg}}\big) \geq 999/1000$, where the probability is over the randomness in Line 23 and $\mathcal{E}_{\mathtt{deg}}$ is defined as follows: For all $a \in [k]$ and $v \in [n]$, we have:*

1. *If $\mathsf{deg}_{\mathcal{G}}(v) \geq 2^{a-w-3}$, then we have $\mathsf{deg}_{\mathcal{G}}(v)/2 < q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v) < 2\mathsf{deg}_{\mathcal{G}}(v)$.*

2. *If $\mathsf{deg}_{\mathcal{G}}(v) \leq 2^{a-w-3}$, then we have $q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v) < 2^{a-w-1}$.*

Proof. We will upper bound the probability that $\mathcal{E}_{\mathtt{deg}}$ does not happen. For this, we fix $a \in [k]$ and $v \in [n]$ and upper bound the probability that one of Items 2 and 3 does not hold for this a and v by $o(1/n^2)$. The proof then follows by a union bound. If $a \leq k^*$, then $\mathsf{deg}_{\mathcal{G}_a}(v) = \mathsf{deg}_{\mathcal{G}}(v)$ and $q_a = 1$ and there is nothing to show. We therefore assume that $a > k^* \Longrightarrow q_a = 2^{k^*-a}$. Note that $\mathsf{deg}_{\mathcal{G}_a}(v)$ is a sum of $\mathsf{deg}_{\mathcal{G}}(v)$-many independent and identically distributed indicator random variables, each of which is 1 with probability q_a.

Consider first the case $\mathsf{deg}_{\mathcal{G}}(v) \geq 2^{a-w-3}$. This means that $q_a\mathsf{deg}_{\mathcal{G}}(v) \geq 2^{k^*-w-3} \geq 50\log n$. In this case, we use Lemmas 2.1 and 2.2 to get:

$$\Pr\big(\mathsf{deg}_{\mathcal{G}_a}(v) \geq 2q_a\mathsf{deg}_{\mathcal{G}}(v)\big) \leq \exp\big(-q_a\mathsf{deg}_{\mathcal{G}}(v)/3\big) = o(1/n^2).$$
$$\Pr\big(\mathsf{deg}_{\mathcal{G}_a}(v) \leq q_a\mathsf{deg}_{\mathcal{G}}(v)/2\big) \leq \exp\big(-q_a\mathsf{deg}_{\mathcal{G}}(v)/8\big) = o(1/n^2).$$

Now, consider the case $\mathsf{deg}_{\mathcal{G}}(v) \leq 2^{a-w-3}$. In this case, we use Corollary 2.4 to get:

$$\Pr\big(\mathsf{deg}_{\mathcal{G}_a}(v) \geq q_a 2^{a-w-1}\big) \leq \exp\big(-q_a 2^{a-w-4}\big) = o(1/n^2).$$

Adding the three bounds gives the result. $\square$

We can deduce several nice properties about the random variable $\widehat{\mathsf{d}\text{-}\mathsf{ind}}^{\mathsf{d}}_a(v)$ defined in Eq. (5.4) when $\mathcal{E}_{\mathtt{deg}}$ happens.

Lemma 5.6. *If $\mathcal{E}_{\mathtt{deg}}$ occurs, the following hold for all $a \in [k]$ and $v \in [n]$ (with probability 1):*

1. *If $\mathsf{deg}_{\mathcal{G}}(v) \leq 2^{a+w+1}$, then we have $\mathsf{deg}_{\mathcal{G}_a}(v) < \texttt{eCutoff}$.*

2. *If* $\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \geq a - w - 1$*, then we have* $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \geq \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) - 1$*.*

3. *We have* $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \leq \max(\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) + 1, a - w - 1)$*.*

Proof. Fix any $\mathcal{G}_{[k]}$ such that $\mathcal{E}_{\mathsf{deg}}$ occurs. Also, fix a and v. Fixing $\mathcal{G}_{[k]}$ also fixes the value of $\mathsf{deg}_{\mathcal{G}_a}(v)$ and $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v)$. We prove each part in turn:

1. If not, we have from Item 2 of Lemma 5.5 that $\mathsf{deg}_{\mathcal{G}}(v) > 2^{a-w-3}$. Item 1 of Lemma 5.5 then implies the following contradiction:
$$\mathtt{eCutoff} \leq \mathsf{deg}_{\mathcal{G}_a}(v) \leq 2q_a\mathsf{deg}_{\mathcal{G}}(v) \leq q_a 2^{a+w+2} \leq 2^{k^*+w+2}.$$

2. Note that $\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \geq a - w - 1$ implies that $\mathsf{deg}_{\mathcal{G}}(v) \geq 2^{a-w-2}$. Item 1 of Lemma 5.5 implies $q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v) > \mathsf{deg}_{\mathcal{G}}(v)/2$. As $d_k \geq \mathsf{deg}_{\mathcal{G}}(v)$ as well, it follows from Eq. (5.4) that $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \geq \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) - 1$.

3. If $q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v) \leq 2^{a-w-1}$, there is nothing to show. Otherwise, we have from Item 2 of Lemma 5.5 that $\mathsf{deg}_{\mathcal{G}}(v) > 2^{a-w-3}$. Now, using Item 1 of Lemma 5.5, we get $q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v) < 2\mathsf{deg}_{\mathcal{G}}(v)$. It follows that $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \leq \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) + 1$.

□

Next, for $a \in [k]$, we define the following set valued random variable that is determined by the randomness in Line 23.
$$\widehat{V}_a = \left\{ v \in [n] \mid \mathsf{deg}_{\mathcal{G}_a}(v) < \mathtt{eCutoff} \wedge \widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \in \mathsf{Win}^{w,k}(a) \right\}. \tag{5.7}$$

We also define the following sets:
$$\begin{aligned} V_a^- &= \left\{ v \in [n] \mid \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \in \mathsf{Win}^{w-1,k}(a) \right\}. \\ V_a^+ &= \left\{ v \in [n] \mid \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \in \mathsf{Win}^{w+1,k}(a) \right\}. \end{aligned} \tag{5.8}$$

Recall that we defined the parameter $D = 2^{k^*+w+2}$. These definitions satisfy:

Lemma 5.9. *If $\mathcal{E}_{\mathsf{deg}}$ occurs, the following hold for all $a \in [k]$ (with probability 1):*

1. *We have* $V_a^- \subseteq \widehat{V}_a \subseteq V_a^+$*.*

2. *For all* $a' \geq a \in [k]$ *and all* $v \in V_{a'}^+$*, we have* $\mathsf{deg}_{\mathcal{G}_a}(v) \leq 2Dq_a/q_{a'}$*.*

3. *For all* $v \in V_a^+$*, we have* $\left| \widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) - \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \right| \leq 1$*.*

Proof. Fix any $\mathcal{G}_{[k]}$ such that $\mathcal{E}_{\mathsf{deg}}$ occurs. Fixing $\mathcal{G}_{[k]}$ also fixes the value of $\mathsf{deg}_{\mathcal{G}_a}(v)$ and $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v)$ for all $a \in [k]$ and $v \in [n]$. Fix $a \in [k]$ as in the lemma. We prove each part in turn.

1. We first show that $V_a^- \subseteq \widehat{V}_a$. Let $v \in V_a^-$ be arbitrary. By Eq. (5.8), we have $\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \in \mathsf{Win}^{w-1,k}(a)$. Using Items 1 to 3 of Lemma 5.6, we get $\mathsf{deg}_{\mathcal{G}_a}(v) < \mathtt{eCutoff}$ and $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \geq \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) - 1 \geq a - w$ and $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \leq \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) + 1 \leq a + w$. It follows from Eq. (5.7) that $v \in \widehat{V}_a$. As $v \in V_a^-$ was arbitrary, we have $V_a^- \subseteq \widehat{V}_a$, as desired.

   We now show that $\widehat{V}_a \subseteq V_a^+$ in the contrapositive. Let $v \in [n] \setminus V_a^+$ be arbitrary. By Eq. (5.8), we have $\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \notin \mathsf{Win}^{w+1,k}(a)$ Thus, either $\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) < a - w - 1$ or $\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) > a + w + 1$. Using Item 3 of Lemma 5.6 in the former case and Item 2 in the latter, we get that either $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \leq a - w - 1 < a - w$ or $\widehat{\mathsf{d}}\text{-}\mathsf{ind}^{\mathbf{d}}_{a}(v) \geq \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) - 1 > a + w$. In either case, we have from Eq. (5.7) that $v \in [n] \setminus \widehat{V}_a$. As $v \in [n] \setminus V_a^+$ was arbitrary, the result follows.

2. As $v \in V_{a'}^+$, we have $\mathsf{d\text{-}ind}_{\mathcal{G}}^{\mathbf{d}}(v) \in \mathsf{Win}^{w+1,k}(a')$. It follows that $\mathsf{d\text{-}ind}_{\mathcal{G}}^{\mathbf{d}}(v) \geq a' - w - 1 \geq a - w - 1$. From Item 3 of Lemma 5.6, we get $\widehat{\mathsf{d\text{-}ind}}_a^{\mathbf{d}}(v) \leq \mathsf{d\text{-}ind}_{\mathcal{G}}^{\mathbf{d}}(v) + 1 \leq a' + w + 2$. From Eq. (5.4), we get that $\min\{q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v), d_k\} \leq 2^{a'+w+2}$. As Item 1 of Lemma 5.5 implies that $q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v) < 2\mathsf{deg}_{\mathcal{G}}(v) \leq 2d_k$, we can continue as $q_a^{-1}\mathsf{deg}_{\mathcal{G}_a}(v) \leq 2^{a'+w+3}$. This means:

$$\mathsf{deg}_{\mathcal{G}_a}(v) \leq 2^{a'+w+3} q_a \leq 2D q_a / q_{a'}.$$

3. As $v \in V_a^+$, we have $\mathsf{d\text{-}ind}_{\mathcal{G}}^{\mathbf{d}}(v) \in \mathsf{Win}^{w+1,k}(a)$. The result follows from Items 2 and 3 of Lemma 5.6.

□

5.3 Biases after edge-subsampling

For $a \in [k]$ and $v \in [n]$, we define a random variable for the *apparent bias index* as follows:

$$\widehat{\mathsf{b\text{-}ind}}_a^{\mathbf{t}}(v) = \mathsf{b\text{-}ind}_{\mathcal{G}_a}^{\mathbf{t}}(v). \tag{5.10}$$

(The right hand side is undefined if $\mathsf{deg}_{\mathcal{G}_a}(v) = 0$.) Note that this random variable is determined by $\mathcal{G}_a$ and therefore, by the randomness in Line 23.

Lemma 5.11. *We have* $\Pr(\mathcal{E}_{\texttt{bias}}) \geq 999/1000$*, where the probability is over the randomness in Line 23 and* $\mathcal{E}_{\texttt{bias}}$ *is defined as follows: For all* $a \in [k]$ *and* $v \in V_a^+$*, we have:*

1. *We have:*
$$\left|\mathsf{deg}_{\mathcal{G}_a}(v) - q_a \mathsf{deg}_{\mathcal{G}}(v)\right| \leq \frac{\epsilon_{\text{bias}}}{12} \cdot q_a \mathsf{deg}_{\mathcal{G}}(v).$$

2. *If* $\mathsf{deg\text{-}out}_{\mathcal{G}}(v) \geq \mathsf{deg}_{\mathcal{G}}(v)/2$*, then we have:*
$$|\mathsf{deg\text{-}out}_{\mathcal{G}_a}(v) - q_a \mathsf{deg\text{-}out}_{\mathcal{G}}(v)| \leq \frac{\epsilon_{\text{bias}}}{12} \cdot q_a \mathsf{deg\text{-}out}_{\mathcal{G}}(v).$$

3. *If* $\mathsf{deg\text{-}in}_{\mathcal{G}}(v) \geq \mathsf{deg}_{\mathcal{G}}(v)/2$*, then we have:*
$$|\mathsf{deg\text{-}in}_{\mathcal{G}_a}(v) - q_a \mathsf{deg\text{-}in}_{\mathcal{G}}(v)| \leq \frac{\epsilon_{\text{bias}}}{12} \cdot q_a \mathsf{deg\text{-}in}_{\mathcal{G}}(v).$$

Proof. We will upper bound the probability that $\mathcal{E}_{\texttt{bias}}$ does not happen. For this, we fix $a \in [k]$ and $v \in V_a^+$ and upper bound the probability that one of Items 1 to 3 does not hold for this a and v by $o(1/n^2)$. The proof then follows by a union bound. If $a \leq k^*$, then $\mathcal{G}_a = \mathcal{G}$ and $q_a = 1$ and there is nothing to show. We therefore assume that $a > k^* \implies q_a = 2^{k^*-a}$ and show Item 2 as the proofs of Items 1 and 3 is analogous. Note that $\mathsf{deg\text{-}out}_{\mathcal{G}_a}(v)$ is a sum of $\mathsf{deg\text{-}out}_{\mathcal{G}}(v)$-many independent and identically distributed indicator random variables, each of which is 1 with probability q_a. Thus, we have from Corollary 2.3 that:

$$\begin{aligned}\Pr\Big(|\mathsf{deg\text{-}out}_{\mathcal{G}_a}(v) - q_a\mathsf{deg\text{-}out}_{\mathcal{G}}(v)| \geq \frac{\epsilon_{\text{bias}}}{12} \cdot q_a\mathsf{deg\text{-}out}_{\mathcal{G}}(v)\Big) &\leq 2\exp\Big(-\frac{\epsilon_{\text{bias}}^2}{500} \cdot q_a\mathsf{deg\text{-}out}_{\mathcal{G}}(v)\Big) \\ &\leq 2\exp\Big(-\frac{\epsilon_{\text{bias}}^2}{1000} \cdot q_a\mathsf{deg}_{\mathcal{G}}(v)\Big) \\ &\leq 2\exp\Big(-\frac{\epsilon_{\text{bias}}^2}{1000} \cdot 2^{k^*-w-2}\Big) \\ &\qquad (\text{Eq. (5.8) implies } \mathsf{deg}_{\mathcal{G}}(v) \geq 2^{a-w-2})\end{aligned}$$

$$= o(1/n^2).$$

□

Lemma 5.12. *If $\mathcal{E}_{\texttt{bias}}$ occurs, for all $a \in [k]$ and $v \in V_a^+$, we have $|\widehat{\mathsf{b}}\text{-}\mathsf{ind}_a^{\mathbf{t}}(v) - \mathsf{b}\text{-}\mathsf{ind}_{\mathcal{G}}^{\mathbf{t}}(v)| \le 1$ (with probability 1).*

Proof. Fix any $\mathcal{G}_{[k]}$ such that $\mathcal{E}_{\texttt{bias}}$ occurs. Also, fix a and v. Fixing $\mathcal{G}_{[k]}$ also fixes the value of $\mathsf{b}\text{-}\mathsf{ind}_{\mathcal{G}}^{\mathbf{t}}(v)$ and $\widehat{\mathsf{b}}\text{-}\mathsf{ind}_a^{\mathbf{t}}(v)$. We assume that $\mathsf{deg}\text{-}\mathsf{out}_{\mathcal{G}}(v) \ge \mathsf{deg}_{\mathcal{G}}(v)/2$ as the proof for the case $\mathsf{deg}\text{-}\mathsf{in}_{\mathcal{G}}(v) \ge \mathsf{deg}_{\mathcal{G}}(v)/2$ is analogous. By Items 1 and 2 of Lemma 5.11, we have:

$$1 - \frac{\epsilon_{\text{bias}}}{12} \le \frac{\mathsf{deg}\text{-}\mathsf{out}_{\mathcal{G}_a}(v)}{q_a \mathsf{deg}\text{-}\mathsf{out}_{\mathcal{G}}(v)} \le 1 + \frac{\epsilon_{\text{bias}}}{12}.$$
$$1 - \frac{\epsilon_{\text{bias}}}{12} \le \frac{\mathsf{deg}_{\mathcal{G}_a}(v)}{q_a \mathsf{deg}_{\mathcal{G}}(v)} \le 1 + \frac{\epsilon_{\text{bias}}}{12}.$$

It follows that
$$1 - \frac{\epsilon_{\text{bias}}}{4} \le \frac{\mathsf{deg}\text{-}\mathsf{out}_{\mathcal{G}_a}(v)/\mathsf{deg}\text{-}\mathsf{out}_{\mathcal{G}}(v)}{\mathsf{deg}_{\mathcal{G}_a}(v)/\mathsf{deg}_{\mathcal{G}}(v)} \le 1 + \frac{\epsilon_{\text{bias}}}{4},$$
which implies
$$\frac{\mathsf{deg}\text{-}\mathsf{out}_{\mathcal{G}}(v)}{\mathsf{deg}_{\mathcal{G}}(v)} - \frac{\epsilon_{\text{bias}}}{4} \le \frac{\mathsf{deg}\text{-}\mathsf{out}_{\mathcal{G}_a}(v)}{\mathsf{deg}_{\mathcal{G}_a}(v)} \le \frac{\mathsf{deg}\text{-}\mathsf{out}_{\mathcal{G}}(v)}{\mathsf{deg}_{\mathcal{G}}(v)} + \frac{\epsilon_{\text{bias}}}{4}.$$
As $\mathsf{bias} = 2\mathsf{deg}\text{-}\mathsf{out}/\mathsf{deg} - 1$, we get:

$$\mathsf{bias}_{\mathcal{G}}(v) - \frac{\epsilon_{\text{bias}}}{2} \le \mathsf{bias}_{\mathcal{G}_a}(v) \le \mathsf{bias}_{\mathcal{G}}(v) + \frac{\epsilon_{\text{bias}}}{2}.$$

As consecutive entries of $\mathbf{t}$ are at least $\epsilon_{\text{bias}}/2$ apart, this means that $|\widehat{\mathsf{b}}\text{-}\mathsf{ind}_a^{\mathbf{t}}(v) - \mathsf{b}\text{-}\mathsf{ind}_{\mathcal{G}}^{\mathbf{t}}(v)| \le 1$ as desired. □

Next, for $a \in [k]$ and $i \in [\ell]$, we define the following set valued random variable that is determined by the randomness in Line 23.

$$\widehat{V}_{a,i} = \left\{ v \in \widehat{V}_a \mid \widehat{\mathsf{b}}\text{-}\mathsf{ind}_a^{\mathbf{t}}(v) \in \mathsf{Win}^{w,\ell}(i) \right\}. \tag{5.13}$$

We also define the following sets for all $a \in [k]$ and $i \in [\ell]$:

$$\begin{aligned} V_{a,i}^- &= \left\{ v \in V_a^- \mid \mathsf{b}\text{-}\mathsf{ind}_{\mathcal{G}}^{\mathbf{t}}(v) \in \mathsf{Win}^{w-1,\ell}(i) \right\}. \\ V_{a,i}^+ &= \left\{ v \in V_a^+ \mid \mathsf{b}\text{-}\mathsf{ind}_{\mathcal{G}}^{\mathbf{t}}(v) \in \mathsf{Win}^{w+1,\ell}(i) \right\}. \end{aligned} \tag{5.14}$$

These definitions satisfy:

Lemma 5.15. *If $\mathcal{E}_{\texttt{deg}}$ and $\mathcal{E}_{\texttt{bias}}$ occur, for all $a \in [k]$ and $i \in [\ell]$, we have $V_{a,i}^- \subseteq \widehat{V}_{a,i} \subseteq V_{a,i}^+$ (with probability 1).*

Proof. Follows immediately from Item 1 of Lemma 5.9, Lemma 5.12, and the definition of Win. □

Recall the set $\texttt{vEst}_{a,i}$ defined in Line 35 and the sets $\mathcal{N}_a$ and S_a defined at the beginning of Section 5. We have:

Lemma 5.16. *Fix any $\mathcal{G}_{[k]}$ such that all of $\mathcal{E}_{\texttt{of1}}$, $\mathcal{E}_{\texttt{deg}}$, and $\mathcal{E}_{\texttt{bias}}$ occur. If $\mathcal{G}_{[k]}$ is sampled in Line 23 and $\mathcal{E}_{\texttt{of2}}^{\mathcal{G}_{[k]}}$ occurs, then the following hold (with probability 1):*

1. *For all $a \in [k]$ and $i \in [\ell]$, we have $\widehat{V}_{a,i} \cap S_a = \texttt{vEst}_{a,i}$.*
2. *Let $a, b \in [k]$ and $i, j \in [\ell]$. For all $u \in \texttt{vEst}_{a,i}$ and $v \in \texttt{vEst}_{b,j}$, it holds that:*

$$\nu^{-w,k,\ell}\Big(\textsf{db-ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\Big) \leq \nu\texttt{Est}^{w,k,\ell}_{a,b}(u,v) \leq \nu^{+w,k,\ell}\Big(\textsf{db-ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\Big).$$

Proof. We fix the randomness in Lines 19 and 23 arbitrarily such that all the events in the lemma occur. Note that fixing the randomness fixes the value of all variables in Algorithms 3 and 4 and also fixes the random variables defined above. We prove each part in turn.

1. Fix $a \in [k]$ and $i \in [\ell]$. Observe that, for any $v \in [n]$, we have the following equivalences:

$$\begin{aligned}
&v \in \widehat{V}_{a,i} \cap S_a\\
&\quad\iff v \in \widehat{V}_{a,i} \wedge v \in S_a\\
&\quad\iff v \in \widehat{V}_a \wedge \widehat{\textsf{b-ind}}^{\mathbf{t}}_a(v) \in \textsf{Win}^{w,\ell}(i) \wedge v \in S_a && \text{(Eq. (5.13))}\\
&\quad\iff \textsf{deg}_{\mathcal{G}_a}(v) < \texttt{eCutoff} \wedge \widehat{\textsf{d-ind}}^{\mathbf{d}}_a(v) \in \textsf{Win}^{w,k}(a) \wedge \widehat{\textsf{b-ind}}^{\mathbf{t}}_a(v) \in \textsf{Win}^{w,\ell}(i) \wedge v \in S_a\\
&&& \text{(Eq. (5.7))}\\
&\quad\iff \textsf{deg}_{\mathcal{G}_a}(v) < \texttt{eCutoff} \wedge \widehat{\textsf{d-ind}}^{\mathbf{d}}_a(v) \in \textsf{Win}^{w,k}(a) \wedge \widehat{\textsf{b-ind}}^{\mathbf{t}}_a(v) \in \textsf{Win}^{w,\ell}(i) \wedge v \in \mathcal{N}_a \cap S_a\\
&&& \text{(As $\widehat{\textsf{d-ind}}^{\mathbf{d}}_a(v) \in \textsf{Win}^{w,k}(a)$ implies by Eq. (5.4) that $\textsf{deg}_{\mathcal{G}_a}(v) > 0 \implies v \in \mathcal{N}_a$)}\\
&\quad\iff \textsf{deg}_{\mathcal{G}_a}(v) < \texttt{eCutoff} \wedge \widehat{\textsf{d-ind}}^{\mathbf{d}}_a(v) \in \textsf{Win}^{w,k}(a) \wedge \widehat{\textsf{b-ind}}^{\mathbf{t}}_a(v) \in \textsf{Win}^{w,\ell}(i) \wedge v \in \texttt{vStored}_a\\
&&& \text{(Item 2 of Lemma 5.3)}\\
&\quad\iff \textsf{deg}_{\texttt{eStored}_a}(v) < \texttt{eCutoff} \wedge \widehat{\textsf{d-ind}}^{\mathbf{d}}_a(v) \in \textsf{Win}^{w,k}(a) \wedge \widehat{\textsf{b-ind}}^{\mathbf{t}}_a(v) \in \textsf{Win}^{w,\ell}(i)\\
&\qquad\qquad \wedge v \in \texttt{vStored}_a.\\
&&& \text{(Item 4 of Lemma 5.3)}
\end{aligned}$$

To continue, note that for any $v \in \texttt{vStored}_a$ such that $\textsf{deg}_{\texttt{eStored}_a}(v) < \texttt{eCutoff}$, we have from Item 3 of Lemma 5.3 that $\textsf{deg-in}_{\texttt{eStored}_a}(v) = \textsf{deg-in}_{\mathcal{G}_a}(v)$ and $\textsf{deg-out}_{\texttt{eStored}_a}(v) = \textsf{deg-out}_{\mathcal{G}_a}(v)$. Conclude from Line 34 and Eqs. (5.4) and (5.10) that for any such v, we have $\widehat{\textsf{d-ind}}^{\mathbf{d}}_a(v) = \textsf{ind}^{\mathbf{d}}(\texttt{dEst}_a(v))$ and $\widehat{\textsf{b-ind}}^{\mathbf{t}}_a(v) = \textsf{ind}^{\mathbf{t}}(\texttt{bEst}_a(v))$. We can now continue as:

$$\begin{aligned}
v \in \widehat{V}_{a,i} \cap S_a &\iff \textsf{deg}_{\texttt{eStored}_a}(v) < \texttt{eCutoff} \wedge \textsf{ind}^{\mathbf{d}}(\texttt{dEst}_a(v)) \in \textsf{Win}^{w,k}(a)\\
&\qquad \wedge \textsf{ind}^{\mathbf{t}}(\texttt{bEst}_a(v)) \in \textsf{Win}^{w,\ell}(i) \wedge v \in \texttt{vStored}_a\\
&\iff v \in \texttt{vEst}_{a,i}. && \text{(Line 35)}
\end{aligned}$$

2. As $u \in \texttt{vEst}_{a,i}$, we have by Line 35 that $u \in \texttt{vStored}_a$ and $\textsf{deg}_{\texttt{eStored}_a}(u) < \texttt{eCutoff}$. From Item 3 of Lemma 5.3, this means that $\textsf{deg-in}_{\texttt{eStored}_a}(u) = \textsf{deg-in}_{\mathcal{G}_a}(u)$ and $\textsf{deg-out}_{\texttt{eStored}_a}(u) = \textsf{deg-out}_{\mathcal{G}_a}(u)$. Using Line 34 and Eqs. (5.4) and (5.10), this means that $\widehat{\textsf{d-ind}}^{\mathbf{d}}_a(u) = \textsf{ind}^{\mathbf{d}}(\texttt{dEst}_a(u))$ and $\widehat{\textsf{b-ind}}^{\mathbf{t}}_a(u) = \textsf{ind}^{\mathbf{t}}(\texttt{bEst}_a(u))$. As similar arguments apply for v, we get from Line 36 that:

$$\nu\texttt{Est}^{w,k,l}_{a,b}(u,v) = \nu^{\sim w,k,\ell}\Big(\widehat{\textsf{d-ind}}^{\mathbf{d}}_a(u), \widehat{\textsf{d-ind}}^{\mathbf{d}}_b(v), \widehat{\textsf{b-ind}}^{\mathbf{t}}_a(u), \widehat{\textsf{b-ind}}^{\mathbf{t}}_b(v)\Big)$$

Now, as we have Definition 3.17, it suffices to show that:

$$\Big(\widehat{\textsf{d-ind}}^{\mathbf{d}}_a(u), \widehat{\textsf{d-ind}}^{\mathbf{d}}_b(v), \widehat{\textsf{b-ind}}^{\mathbf{t}}_a(u), \widehat{\textsf{b-ind}}^{\mathbf{t}}_b(v)\Big) \in \textsf{Win}^{1,k,\ell}\Big(\textsf{db-ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\Big).$$

From Definition 3.10 and Eq. (3.5), this follows if we show that:

$$\left|\widehat{\mathsf{d}\text{-}\mathsf{ind}}^{\mathbf{d}}_{a}(u) - \mathsf{d}\text{-}\mathsf{ind}^{\mathbf{d}}_{\mathcal{G}}(u)\right|, \left|\widehat{\mathsf{d}\text{-}\mathsf{ind}}^{\mathbf{d}}_{b}(v) - \mathsf{d}\text{-}\mathsf{ind}^{\mathbf{d}}_{\mathcal{G}}(v)\right| \le 1,$$
$$\left|\widehat{\mathsf{b}\text{-}\mathsf{ind}}^{\mathbf{t}}_{a}(u) - \mathsf{b}\text{-}\mathsf{ind}^{\mathbf{t}}_{\mathcal{G}}(u)\right|, \left|\widehat{\mathsf{b}\text{-}\mathsf{ind}}^{\mathbf{t}}_{b}(v) - \mathsf{b}\text{-}\mathsf{ind}^{\mathbf{t}}_{\mathcal{G}}(v)\right| \le 1.$$

This first inequality is due to Items 2 and 3 of Lemma 5.6 while the second inequality is due to Lemma 5.12 (note that Item 1 implies that $u \in \widehat{V}_{a,i}$ and $v \in \widehat{V}_{b,j}$ and we have Lemma 5.15).

□

5.4 Counting target edges after edge-subsampling

For $a, b \in [k]$ and $i, j \in [\ell]$, define:

$$E^{-}_{a,b,i,j} = \mathcal{G} \cap \left(V^{-}_{a,i} \times V^{-}_{b,j}\right) \qquad \text{and} \qquad E^{+}_{a,b,i,j} = \mathcal{G} \cap \left(V^{+}_{a,i} \times V^{+}_{b,j}\right). \tag{5.17}$$

Recall from the statement of Lemma 4.4 the notation $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$. Next, for $a, b \in [k]$ and $i, j \in [\ell]$, we define the random variables:

$$\begin{aligned} Y^{-}_{a,b,i,j} &= \sum_{(u,v)\in E^{-}_{a,b,i,j}} \mathbb{1}_{(u,v)\in\mathcal{G}_{\min\{a,b\}}} \nu^{-w,k,\ell}\left(\mathsf{db}\text{-}\mathsf{ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\right). \\ Y^{+}_{a,b,i,j} &= \sum_{(u,v)\in E^{+}_{a,b,i,j}} \mathbb{1}_{(u,v)\in\mathcal{G}_{\min\{a,b\}}} \nu^{+w,k,\ell}\left(\mathsf{db}\text{-}\mathsf{ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\right). \end{aligned} \tag{5.18}$$

Recall the notation in Definition 3.18. The following lemma calculates the expectation of these random variables.

Lemma 5.19. *For all $a, b \in [k]$ and $i, j \in [\ell]$, it holds that:*

$$\mathbb{E}[Y^{-}_{a,b,i,j}] = q_{\min\{a,b\}} m \cdot A^{-w}(a,b,i,j) \qquad \text{and} \qquad \mathbb{E}[Y^{+}_{a,b,i,j}] = q_{\min\{a,b\}} m \cdot A^{+w}(a,b,i,j).$$

Proof. We only show the first equation as the proof of the second one is analogous. Note from Eq. (5.18) and linearity of expectation that it suffices to show that:

$$m \cdot A^{-w}(a,b,i,j) = \sum_{(u,v)\in E^{-}_{a,b,i,j}} \nu^{-w,k,\ell}\left(\mathsf{db}\text{-}\mathsf{ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\right). \tag{5.20}$$

This is because:

$$\begin{aligned} m \cdot A^{-w}(a,b,i,j) &= m \cdot \sum_{(a',b',i',j')\in\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)} \nu^{-w,k,\ell}(a',b',i',j') A(a',b',i',j') \quad \text{(Definition 3.18)} \\ &= \sum_{(a',b',i',j')\in\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)} \nu^{-w,k,\ell}(a',b',i',j') \sum_{(u,v)\in\mathcal{G}} \mathbb{1}_{\mathsf{db}\text{-}\mathsf{ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)=(a',b',i',j')} \\ &\hspace{30em} \text{(Eq. (3.7))} \\ &= \sum_{(u,v)\in\mathcal{G}} \mathbb{1}_{\mathsf{db}\text{-}\mathsf{ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\in\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)} \cdot \nu^{-w,k,\ell}\left(\mathsf{db}\text{-}\mathsf{ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\right). \end{aligned}$$

Now, by Eq. (5.17), it suffices to show that, for all $(u,v) \in \mathcal{G}$, we have:

$$\mathsf{db\text{-}ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v) \in \mathsf{Win}^{w-1,k,\ell}(a,b,i,j) \iff (u,v) \in V^-_{a,i} \times V^-_{b,j}.$$

This is because:

$$\begin{aligned}
&\mathsf{db\text{-}ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v) \in \mathsf{Win}^{w-1,k,\ell}(a,b,i,j) \\
&\quad\iff \left(\mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(u), \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v), \mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u), \mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v)\right) \in \mathsf{Win}^{w-1,k,\ell}(a,b,i,j) && \text{(Eq. (3.5))} \\
&\quad\iff \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(u) \in \mathsf{Win}^{w-1,k}(a) \wedge \mathsf{d\text{-}ind}^{\mathbf{d}}_{\mathcal{G}}(v) \in \mathsf{Win}^{w-1,k}(b) \\
&\qquad\qquad \wedge \mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u) \in \mathsf{Win}^{w-1,\ell}(i) \wedge \mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v) \in \mathsf{Win}^{w-1,\ell}(j) && \text{(Definition 3.10)} \\
&\quad\iff u \in V^-_{a,i} \wedge v \in V^-_{b,j}. && \text{(Eqs. (5.8) and (5.14))}
\end{aligned}$$

□

Using standard concentration bounds, we get:

Lemma 5.21. *We have* $\Pr(\mathcal{E}_{\texttt{count1}}) \geq 999/1000$*, where the probability is over the randomness in Line 23 and* $\mathcal{E}_{\texttt{count1}}$ *is defined as follows: For all* $a,b \in [k]$ *and* $i,j \in [\ell]$*, we have:*

$$\left|Y^-_{a,b,i,j} - \mathbb{E}[Y^-_{a,b,i,j}]\right| \leq \frac{\epsilon'}{2(k\ell)^2} \cdot q_{\min\{a,b\}} m.$$
$$\left|Y^+_{a,b,i,j} - \mathbb{E}[Y^+_{a,b,i,j}]\right| \leq \frac{\epsilon'}{2(k\ell)^2} \cdot q_{\min\{a,b\}} m.$$

Proof. We will upper bound the probability that $\mathcal{E}_{\texttt{count1}}$ does not happen. For this, we fix $a,b \in [k]$ and $i,j \in [\ell]$ and upper bound the probability that the first inequality does not hold for this a,b and i,j by $o(1/n)$. The proof for the second inequality is similar and the lemma, thus follows by a union bound. Fix $a,b \in [k]$ and $i,j \in [\ell]$. We have from Lemma 5.19 and Corollary 2.6 that:

$$\begin{aligned}
\Pr\left(\left|Y^-_{a,b,i,j} - \mathbb{E}[Y^-_{a,b,i,j}]\right| \geq \frac{\epsilon'}{2(k\ell)^2} \cdot q_{\min\{a,b\}} m\right) &\leq 2\exp\left(-\frac{\epsilon'^2 \cdot q_{\min\{a,b\}} m}{12(k\ell)^4 A^{-w}(a,b,i,j)}\right) \\
&\leq 2\exp\left(-\frac{\epsilon'^2 \cdot q_{\min\{a,b\}} m}{12(k\ell)^4}\right) && \text{(Lemma 3.19)} \\
&\leq 2\exp\left(-\frac{\epsilon'^2 q_k m}{12(k\ell)^4}\right) && \text{(Table 4)} \\
&= o(1/n),
\end{aligned}$$

where, for the last step, recall that $k = \log(2\widehat{m}) = O_\epsilon(\log n)$, while $q_k m = 2^{k^*-k} m = m \cdot \frac{\log^6 n}{2\widehat{m}} \geq 0.5 \log^6 n$. So the negated expression in the exponent is $\Theta_\epsilon(\log^2 n)$ and we get the required bound of $o(1/n)$. □

5.5 Counting target edges after vertex-subsampling

Recall that $S_a = \{v \in [n] : \pi_a(v) = 1\}$ for $a \in [k]$ and that S_a is determined by the randomness in Line 19. Observe from Line 26 that only vertices in S_a can possibly be stored in $\texttt{vStored}_a$. Next, let $\mathcal{G}_1, \ldots, \mathcal{G}_k$ be a set of subgraphs of $\mathcal{G}$ that may be sampled in Line 23 and recall our notation $\mathcal{G}_{[k]} = \mathcal{G}_1, \ldots, \mathcal{G}_k$. For such a $\mathcal{G}_{[k]}$ and any $a, b \in [k]$ and $i, j \in [\ell]$, define the random variable:

$$\begin{aligned} X_{a,b,i,j}^{\mathcal{G}_{[k]},-} &= \sum_{(u,v) \in E^-_{a,b,i,j} \cap \mathcal{G}_{\min\{a,b\}}} \mathbb{1}_{u \in S_a} \mathbb{1}_{v \in S_b} \nu^{-w,k,\ell}\Big(\textsf{db-ind}_{\mathcal{G}}^{\mathbf{d},\mathbf{t}}(u,v)\Big). \\ X_{a,b,i,j}^{\mathcal{G}_{[k]},+} &= \sum_{(u,v) \in E^+_{a,b,i,j} \cap \mathcal{G}_{\min\{a,b\}}} \mathbb{1}_{u \in S_a} \mathbb{1}_{v \in S_b} \nu^{+w,k,\ell}\Big(\textsf{db-ind}_{\mathcal{G}}^{\mathbf{d},\mathbf{t}}(u,v)\Big). \end{aligned} \tag{5.22}$$

Note that the random variables $X_{a,b,i,j}^{\mathcal{G}_{[k]},-}$ and $X_{a,b,i,j}^{\mathcal{G}_{[k]},+}$ are determined solely by the randomness in Line 19 (as they only depend on the randomness in the sets $S_1, \ldots, S_k$). Our definitions satisfy the concentration lemma in Lemma 5.24 below but first we calculate the expectation of the random variables defined above (recall that the value of $Y^-_{a,b,i,j}$ and $Y^+_{a,b,i,j}$, for all $a, b \in [k]$ and $i, j \in [\ell]$, and whether or not $\mathcal{E}_{\texttt{deg}}$ and $\mathcal{E}_{\texttt{count1}}$ occur is determined by the graphs $\mathcal{G}_{[k]}$):

Lemma 5.23. *Fix any $\mathcal{G}_{[k]}$. For all $a, b \in [k]$ and $i, j \in [\ell]$, we have:*

$$\mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},-}] = p_a p_b Y^-_{a,b,i,j} \qquad \text{and} \qquad \mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},+}] = p_a p_b Y^+_{a,b,i,j}.$$

Proof. We only prove the former as the latter is analogous. By linearity of expectation, Eq. (5.18) and the fact that the hash-functions sampled in Line 19 are 4-wise independent, we have:

$$\mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},-}] = \sum_{(u,v) \in E^-_{a,b,i,j} \cap \mathcal{G}_{\min\{a,b\}}} p_a p_b \cdot \nu^{-w,k,\ell}(\textsf{db-ind}_{\mathcal{G}}^{\mathbf{d},\mathbf{t}}(u,v)) = p_a p_b \cdot Y^-_{a,b,i,j}.$$

□

Lemma 5.24. *Fix any $\mathcal{G}_{[k]}$ such that both $\mathcal{E}_{\texttt{deg}}$ and $\mathcal{E}_{\texttt{count1}}$ occur. It holds that* $\Pr\Big(\mathcal{E}_{\texttt{count2}}^{\mathcal{G}_{[k]}}\Big) \geq 999/1000$, *where the probability is over the randomness in Line 19 and $\mathcal{E}_{\texttt{count2}}^{\mathcal{G}_{[k]}}$ is defined as follows: For all $a, b \in [k]$ and $i, j \in [\ell]$, we have:*

$$\begin{aligned} \Big|X_{a,b,i,j}^{\mathcal{G}_{[k]},-} - \mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},-}]\Big| &\leq \frac{\epsilon'}{2(k\ell)^2} \cdot p_a p_b q_{\min\{a,b\}} m. \\ \Big|X_{a,b,i,j}^{\mathcal{G}_{[k]},+} - \mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},+}]\Big| &\leq \frac{\epsilon'}{2(k\ell)^2} \cdot p_a p_b q_{\min\{a,b\}} m. \end{aligned}$$

Proof. Fix $\mathcal{G}_{[k]}$ and note that this also fixes $Y^-_{a,b,i,j}$ and $Y^+_{a,b,i,j}$. We will upper bound the probability that $\mathcal{E}_{\texttt{count2}}^{\mathcal{G}_{[k]}}$ does not happen. For this, we fix $a, b \in [k]$ and $i, j \in [\ell]$ and upper bound the probability that the first inequality does not hold for this a, b and i, j by $\frac{10^{-4}}{(k\ell)^2}$. The proof for the second inequality is similar and the lemma, thus follows by a union bound. Fix an arbitrary $a, b \in [k]$ and $i, j \in [\ell]$ and assume $a \leq b$ without loss of generality.

Note from Table 4 that $p_0 \leq p_1 \cdots \leq p_k \leq 1$. This means that if $p_a = 1$, then $p_a = p_b = 1$ and $X_{a,b,i,j}^{\mathcal{G}_{[k]},-}$ is a constant value independent of the randomness and the inequality follows. Thus, we can assume $p_a < 1$ which means that $p_a = p_0 q_a^{-1}$. For convenience, define $F = E^-_{a,b,i,j} \cap \mathcal{G}_a$ and $\nabla = \frac{\epsilon'}{2(k\ell)^2} \cdot p_a p_b q_a m$. For $(u, v) \in F$, define the random variable:

$$I_{(u,v)} = \mathbb{1}_{u \in S_a} \mathbb{1}_{v \in S_b} \nu^{-w,k,\ell}\Big(\textsf{db-ind}_{\mathcal{G}}^{\mathbf{d},\mathbf{t}}(u,v)\Big).$$

Plugging into Eq. (5.22), we get that $X_{a,b,i,j}^{\mathcal{G}_{[k]},-} = \sum_{(u,v)\in F} I_{(u,v)}$. Next, we claim that for all $(u,v) \in F$, there are at most $5D \cdot \frac{p_b}{p_0}$ many $(u',v') \in F$ such that $I_{(u,v)}$ and $I_{(u',v')}$ are not independent, where D and p_0 are as in Tables 2 and 3. Assuming this claim for now, Corollary 2.8 says:

$$\Pr\left(\left|X_{a,b,i,j}^{\mathcal{G}_{[k]},-} - \mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},-}]\right| \geq \nabla\right) \leq \frac{5Dp_b \cdot \mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},-}]}{p_0\nabla^2}.$$

Now, recall from Lemma 5.23 that $\mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},-}] = p_a p_b Y_{a,b,i,j}^-$. As $\mathcal{E}_{\texttt{count1}}$ occurs, we have from Lemmas 5.19 and 5.21 that $Y_{a,b,i,j}^- \leq q_a m \cdot \left(A^{-w}(a,b,i,j) + \frac{\epsilon'}{2(k\ell)^2}\right) \leq 2q_a m$ by Lemma 3.19. Plugging in we get:

$$\Pr\left(\left|X_{a,b,i,j}^{\mathcal{G}_{[k]},-} - \mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},-}]\right| \geq \nabla\right) \leq \frac{10Dp_b \cdot p_a p_b q_a m}{p_0\nabla^2} = \frac{40(k\ell)^2 D}{p_0\epsilon'^2 p_a q_a m},$$

by definition of ∇. Recalling that $p_a = p_0 q_a^{-1}$, we get:

$$\Pr\left(\left|X_{a,b,i,j}^{\mathcal{G}_{[k]},-} - \mathbb{E}[X_{a,b,i,j}^{\mathcal{G}_{[k]},-}]\right| \geq \nabla\right) \leq \frac{80(k\ell)^2 D}{\rho^2\epsilon'^2} \leq \frac{10^{-4}}{(k\ell)^2}.$$

It remains to show the claim. For this, we consider two cases, based on whether or not $p_b = 1$. In both cases, we use the fact that the hash-functions sampled in Line 19 are 4-wise independent. If $p_b = 1$, then $S_b = [n]$ (with probability 1) and $I_{(u,v)}$ and $I_{(u',v')}$ are not independent only if $u = u'$. As $F \subseteq \mathcal{G}_a$, this means that the number of $(u',v') \in F$ such that the random variables $I_{(u,v)}$ and $I_{(u',v')}$ are not independent is at most $\mathsf{deg}_{\mathcal{G}_a}(u)$, implying that it suffices to show the bound $\mathsf{deg}_{\mathcal{G}_a}(u) \leq 2D$. For this, note that $F \subseteq E_{a,b,i,j}^-$ implies by Eq. (5.17) that $u \in V_{a,i}^- \subseteq V_a^+$ by Eq. (5.14) and Item 1 of Lemma 5.9. Now, using Item 2 of Lemma 5.9, we get that $\mathsf{deg}_{\mathcal{G}_a}(u) \leq 2D$, as desired.

Now, consider the case $p_b = 1$ which implies $p_b/p_0 = 1/q_b$. In this case, we have that $I_{(u,v)}$ and $I_{(u',v')}$ are not independent only if the pairs (u,v) and (u',v') have a common element. Using the fact that $F \subseteq \mathcal{G}_a$, we get that the number of $(u',v') \in F$ such that the random variables $I_{(u,v)}$ and $I_{(u',v')}$ are not independent is at most $2 + \mathsf{deg}_{\mathcal{G}_a}(u) + \mathsf{deg}_{\mathcal{G}_a}(v)$, implying that it suffices to show the bounds $\mathsf{deg}_{\mathcal{G}_a}(u) \leq 2D$ and $\mathsf{deg}_{\mathcal{G}_a}(v) \leq 2D/q_b$. For this, note that $F \subseteq E_{a,b,i,j}^-$ implies by Eq. (5.17) that $u \in V_{a,i}^-$ and $v \in V_{b,j}^-$. As in the previous part, we get that $u \in V_a^+$ and $v \in V_b^+$. Now, using Item 2 of Lemma 5.9, we get that $\mathsf{deg}_{\mathcal{G}_a}(u) \leq 2D$ and $\mathsf{deg}_{\mathcal{G}_a}(v) \leq 2D/q_b$, as desired. □

5.6 Putting it all together: Proving Lemma 4.4

We are now ready to prove Lemma 4.4.

Proof of Lemma 4.4. Define the event $\mathcal{E}_{\mathsf{good}}$ (over the randomness in Lines 19 and 23) that for all $a, b \in [k]$ and $i, j \in [\ell]$, it holds that:

$$A^{-w}(a,b,i,j) - \frac{\epsilon'}{(k\ell)^2} \leq \widehat{A}(a,b,i,j) \leq A^{+w}(a,b,i,j) + \frac{\epsilon'}{(k\ell)^2}.$$

Observe that Lemma 4.4 is the same as showing $\Pr(\mathcal{E}_{\mathsf{good}}) \geq 99/100$. We show this by upper bounding the probability of the complement event. Using a union bound, we have:

$$\Pr(\overline{\mathcal{E}_{\mathsf{good}}}) \leq \Pr(\overline{\mathcal{E}_{\texttt{of1}}} \vee \overline{\mathcal{E}_{\texttt{deg}}} \vee \overline{\mathcal{E}_{\texttt{bias}}} \vee \overline{\mathcal{E}_{\texttt{count1}}}) + \Pr(\overline{\mathcal{E}_{\mathsf{good}}} \mid \mathcal{E}_{\texttt{of1}}, \mathcal{E}_{\texttt{deg}}, \mathcal{E}_{\texttt{bias}}, \mathcal{E}_{\texttt{count1}})$$

$$\leq \frac{4}{1000} + \Pr\left(\overline{\mathcal{E}_{\texttt{good}}} \mid \mathcal{E}_{\texttt{of1}}, \mathcal{E}_{\texttt{deg}}, \mathcal{E}_{\texttt{bias}}, \mathcal{E}_{\texttt{count1}}\right). \qquad \text{(Lemmas 5.1, 5.5, 5.11 and 5.21)}$$

Thus, it suffices to bound the last term by $\frac{2}{1000}$. For this, note that the events $\mathcal{E}_{\texttt{of1}}$, $\mathcal{E}_{\texttt{deg}}$, $\mathcal{E}_{\texttt{bias}}$, and $\mathcal{E}_{\texttt{count1}}$ are all determined by the randomness in Line 23, or equivalently by the graphs $\mathcal{G}_{[k]} = \mathcal{G}_1, \ldots, \mathcal{G}_k$. Thus, it suffices to show that for any $\mathcal{G}_{[k]}$ such that all the events $\mathcal{E}_{\texttt{of1}}$, $\mathcal{E}_{\texttt{deg}}$, $\mathcal{E}_{\texttt{bias}}$, and $\mathcal{E}_{\texttt{count1}}$ occur, we have $\Pr\left(\overline{\mathcal{E}_{\texttt{good}}} \mid \mathcal{G}_{[k]}\right) \leq \frac{1}{1000}$. Fix such a $\mathcal{G}_{[k]}$. By another union bound, we have:

$$\begin{aligned}\Pr\left(\overline{\mathcal{E}_{\texttt{good}}} \mid \mathcal{G}_{[k]}\right) &\leq \Pr\left(\overline{\mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{of2}}} \vee \overline{\mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{count2}}} \mid \mathcal{G}_{[k]}\right) + \Pr\left(\overline{\mathcal{E}_{\texttt{good}}} \mid \mathcal{G}_{[k]}, \mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{of2}}, \mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{count2}}\right) \\ &\leq \frac{2}{1000} + \Pr\left(\overline{\mathcal{E}_{\texttt{good}}} \mid \mathcal{G}_{[k]}, \mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{of2}}, \mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{count2}}\right), \qquad \text{(Lemmas 5.2 and 5.24)}\end{aligned}$$

where the last inequality also uses the fact that $\mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{count2}}$ and $\mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{of2}}$ are determined by the randomness in Line 19 and therefore, independent of $\mathcal{G}_{[k]}$. To finish the proof, we show that for any choice of the randomness in Lines 19 and 23 such that the events $\mathcal{G}_{[k]}$, $\mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{of2}}$, and $\mathcal{E}^{\mathcal{G}_{[k]}}_{\texttt{count2}}$ occur, the event $\mathcal{E}_{\texttt{good}}$ also occurs. Fix such a randomness for the rest of the proof. Note that fixing this randomness fixes the value of all the random variables defined above and also fixes the value of all variables in Algorithms 3 and 4. Henceforth, we shall often abuse notation and use the name of the random variable to denote the value it is fixed to, e.g., $X^{\mathcal{G}_{[k]},-}_{a,b,i,j}$ will denote the value $X^{\mathcal{G}_{[k]},-}_{a,b,i,j}$ is fixed to. We use a similar notation for variables. We first claim that, for all $a, b \in [k]$ and $i, j \in [\ell]$, we have:

$$X^{\mathcal{G}_{[k]},-}_{a,b,i,j} \leq \widehat{A}(a,b,i,j) \cdot p_a p_b q_{\min\{a,b\}} m \leq X^{\mathcal{G}_{[k]},+}_{a,b,i,j}. \tag{5.25}$$

We prove Eq. (5.25) later but assuming it for now, we get from Lemmas 5.23 and 5.24 that (for all $a, b \in [k]$ and $i, j \in [\ell]$):

$$\frac{Y^-_{a,b,i,j}}{q_{\min\{a,b\}} m} - \frac{\epsilon'}{2(k\ell)^2} \leq \widehat{A}(a,b,i,j) \leq \frac{Y^+_{a,b,i,j}}{q_{\min\{a,b\}} m} + \frac{\epsilon'}{2(k\ell)^2}.$$

Continuing using Lemmas 5.19 and 5.21, we get (for all $a, b \in [k]$ and $i, j \in [\ell]$):

$$A^{-w}(a,b,i,j) - \frac{\epsilon'}{(k\ell)^2} \leq \widehat{A}(a,b,i,j) \leq A^{+w}(a,b,i,j) + \frac{\epsilon'}{(k\ell)^2},$$

as desired. It remains to prove Eq. (5.25). We only show the second inequality as the proof for the first one is analogous. For this, fix $a, b \in [k]$ and $i, j \in [\ell]$ and note from Eq. (4.3) that $\widehat{A}(a,b,i,j) \cdot p_a p_b q_{\min\{a,b\}} m = \texttt{AEst}_{a,b,i,j}$. Thus, it suffices to show that $\texttt{AEst}_{a,b,i,j} \leq X^{\mathcal{G}_{[k]},+}_{a,b,i,j}$. For this, we first combine Line 36 with Item 2 of Lemma 5.16 to get:

$$\texttt{AEst}_{a,b,i,j} \leq \sum_{(u,v) \in \texttt{eStored}_{\min\{a,b\}} \cap (\texttt{vEst}_{a,i} \times \texttt{vEst}_{b,j})} \nu^{+w,k,\ell}\left(\textsf{db-ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\right).$$

To continue, we claim that $\texttt{eStored}_{\min\{a,b\}} \cap (\texttt{vEst}_{a,i} \times \texttt{vEst}_{b,j}) = \mathcal{G}_{\min\{a,b\}} \cap (\texttt{vEst}_{a,i} \times \texttt{vEst}_{b,j})$. As the $\subseteq$ part is trivial, it suffices to show that $\supseteq$ part. Fix any $(u,v) \in \mathcal{G}_{\min\{a,b\}} \cap (\texttt{vEst}_{a,i} \times \texttt{vEst}_{b,j})$. Applying Item 3 of Lemma 5.3 implies that $(u,v) \in \texttt{eStored}_{\min\{a,b\}}$ and we are done. With this claim, we get:

$$\texttt{AEst}_{a,b,i,j} \leq \sum_{(u,v) \in \mathcal{G}_{\min\{a,b\}} \cap (\texttt{vEst}_{a,i} \times \texttt{vEst}_{b,j})} \nu^{+w,k,\ell}\left(\textsf{db-ind}^{\mathbf{d},\mathbf{t}}_{\mathcal{G}}(u,v)\right).$$

Next, applying Item 1 of Lemma 5.16 to get $\texttt{vEst}_{a,i} \times \texttt{vEst}_{b,j} = \left(\widehat{V}_{a,i} \cap S_a\right) \times \left(\widehat{V}_{b,j} \cap S_b\right)$. The latter equals $\left(\widehat{V}_{a,i} \times \widehat{V}_{b,j}\right) \cap (S_a \times S_b)$ and we get:

$$\texttt{AEst}_{a,b,i,j} \leq \sum_{(u,v) \in \left(\widehat{V}_{a,i} \times \widehat{V}_{b,j}\right) \cap \mathcal{G}_{\min\{a,b\}}} \mathbb{1}_{u \in S_a} \mathbb{1}_{v \in S_b} \nu^{+w,k,\ell}\left(\textsf{db-ind}_{\mathcal{G}}^{\mathbf{d},\mathbf{t}}(u,v)\right).$$

To finish the proof, use Eqs. (5.17) and (5.22) and Lemma 5.15 to get

$$\texttt{AEst}_{a,b,i,j} \leq \sum_{(u,v) \in E^{+}_{a,b,i,j} \cap \mathcal{G}_{\min\{a,b\}}} \mathbb{1}_{u \in S_a} \mathbb{1}_{v \in S_b} \nu^{+w,k,\ell}\left(\textsf{db-ind}_{\mathcal{G}}^{\mathbf{d},\mathbf{t}}(u,v)\right) = X^{\mathcal{G}_{[k]},+}_{a,b,i,j}.$$

□

6 Analysis of the reduction: Proving Lemma 3.22

In this section, we prove several remaining lemmas from Section 3, which together will prove Lemma 3.22.

6.1 Basic properties of windows

We begin by stating a number of basic facts about windows (presented only for the necessary dimensions, for brevity).

Fact 6.1 (Size of 1D and 4D windows)**.** *The size of d-dimensional windows is bounded within a factor of 2^d. In particular,*

- *In one dimension, for every $w < \ell \in \mathbb{N}$ and $i \in [\ell]$,*

$$w + 1 \leq |\textsf{Win}^{w,\ell}(i)| \leq 2w + 1.$$

- *In four dimensions, for every $w < k, \ell \in \mathbb{N}$ and $a, b \in [k]$, $i, j \in [\ell]$,*

$$(w+1)^4 \leq |\textsf{Win}^{w,k,\ell}(a,b,i,j)| \leq (2w+1)^4.$$

Fact 6.2 (Size difference of 1D windows)**.** *For every $w \leq w' < \ell \in \mathbb{N}$ and $i \in [\ell]$,*

$$|\textsf{Win}^{w',\ell}(i) \setminus \textsf{Win}^{w,\ell}(i)| \leq 2(w' - w).$$

Fact 6.3 (Symmetry of containment)**.** *Containment in windows is a symmetric property, i.e.,*

- *In two dimensions, for every $w < \ell \in \mathbb{N}$ and $i, j, i', j' \in [\ell]$,*

$$(i', j') \in \textsf{Win}^{w,\ell}(i,j) \iff (i,j) \in \textsf{Win}^{w,\ell}(i',j').$$

- *In four dimensions, for every $w < k, \ell \in \mathbb{N}$ and $a, b, a', b' \in [k]$, $i, j, i', j' \in [\ell]$,*

$$(a', b', i', j') \in \textsf{Win}^{w,k,\ell}(a,b,i,j) \iff (a,b,i,j) \in \textsf{Win}^{w,k,\ell}(a',b',i',j').$$

As a corollary, we have the following fact:

Fact 6.4 (Counting containments). *The number of windows a particular index is contained equals the size of its window, i.e.,*

- *In two dimensions, for every $w < \ell \in \mathbb{N}$ and $i', j' \in [\ell]$, the number of distinct windows containing (i', j') is*

$$|\{(i,j) \in [\ell]^2 : \mathsf{Win}^{w,\ell}(i,j) \ni (i',j')\}| = |\mathsf{Win}^{w,\ell}(i',j')|.$$

- *In four dimensions, for every $w < k, \ell \in \mathbb{N}$ and $a', b' \in [k]$, $i', j' \in [\ell]$, the number of distinct windows containing (a', b', i', j') is*

$$|\{(a,b,i,j) \in [k]^2 \times [\ell]^2 : \mathsf{Win}^{w,k,\ell}(a,b,i,j) \ni (a',b',i',j')\}| = |\mathsf{Win}^{w,k,\ell}(a',b',i',j')|.$$

The following is essentially the triangle inequality for the ∞-norm, and roughly states that "nearby windows look alike":

Fact 6.5 (Triangle inequality for windows). *For every $w + w' < k, \ell \in \mathbb{N}$, and every $a, b, a', b' \in [k]$, $i, j, i', j' \in [\ell]$,*

$$(a',b',i',j') \in \mathsf{Win}^{w',k,\ell}(a,b,i,j) \implies \mathsf{Win}^{w,k,\ell}(a',b',i',j') \subseteq \mathsf{Win}^{w+w',k,\ell}(a,b,i,j).$$

6.2 Double-counting arguments: Proving Propositions 3.12 and 3.16

Proof of Proposition 3.12. We claim that the sum of entries in $M^{\sim w}$ is the same as in M. Indeed, this holds by Fact 6.4 and "double-counting". In particular,

$$\begin{aligned}
\sum_{i,j=1}^{\ell} M^{\sim w}(i',j') &= \sum_{i,j=1}^{\ell} \sum_{i',j' \in \mathsf{Win}^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j') M(i',j') && \text{(def. of } M^{\sim w}\text{)} \\
&= \sum_{i',j'=1}^{\ell} |\{(i,j) \in [\ell]^2 : \mathsf{Win}^{w,\ell}(i,j) \ni (i',j')\}| \cdot \nu^{\sim w,\ell}(i',j') M(i',j') \\
& && \text{(exchanging sums)} \\
&= \sum_{i',j'=1}^{\ell} |\mathsf{Win}^{w,\ell}(i',j')| \cdot \nu^{\sim w,\ell}(i',j') M(i',j') && \text{(Fact 6.4)} \\
&= \sum_{i',j'=1}^{\ell} M(i',j'). && \text{(def. of } \nu^{\sim w,\ell}\text{)}
\end{aligned}$$

□

Proof of Proposition 3.16. Recall $M = \mathsf{Proj}(A)$ and $N = \mathsf{Proj}(A^{\sim w})$. We first want to show $M^{\sim w} = N$. Again, we can do this by "double-counting". Recall that $\mathsf{Win}^{w,k,\ell}(a,b,i,j) = \mathsf{Win}^{w,k}(a,b) \times \mathsf{Win}^{w,\ell}(i,j)$ (and thus $\nu^{\sim w,k,\ell}(a,b,i,j) = \nu^{\sim w,k}(a,b) \cdot \nu^{\sim w,\ell}(i,j)$). We have:

$$N(i,j) = \sum_{a,b=1}^{k} A^{\sim w}(a,b,i,j) \qquad \text{(def. of } N\text{)}$$

$$\begin{aligned}
&= \sum_{a,b=1}^{k} \sum_{(a',b',i',j') \in \mathsf{Win}^{w,k,\ell}(a,b,i,j)} \nu^{\sim w,k,\ell}(a',b',i',j') A(a',b',i',j') && \text{(def. of } A^{\sim w}\text{)} \\
&= \sum_{(i',j') \in \mathsf{Win}^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j') \left(\sum_{a,b=1}^{k} \sum_{(a',b') \in \mathsf{Win}^{w,k}(a,b)} \nu^{\sim w,k}(a',b') A(a',b',i',j') \right) && \text{(exchanging sums)} \\
&= \sum_{(i',j') \in \mathsf{Win}^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j') \left(\sum_{a',b'=1}^{k} |\{(a,b) \in [k]^2 : \mathsf{Win}^{w,k}(a,b) \ni (a',b')\}| \cdot \nu^{\sim w,k}(a',b') A(a',b',i',j') \right) && \text{(exchanging sums)} \\
&= \sum_{(i',j') \in W^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j') \left(\sum_{a',b'=1}^{k} A(a',b',i',j') \right) && \text{(Fact 6.4)} \\
&= \sum_{(i',j') \in W^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j') M(i',j') && \text{(def. of } M\text{)} \\
&= M^{\sim w}(i,j). && \text{(def. of } M^{\sim w}\text{)}
\end{aligned}$$

Now, consider an array $\widehat{A} \in \mathbb{A}^{k,\ell}$; recall, $\widehat{M} = \mathsf{Proj}(\widehat{A})$, and we want to show that $\|\widehat{M} - M^{\sim w}\|_1 \leq \|\widehat{A} - A^{\sim w}\|_1$. This follows essentially from the triangle inequality. Indeed:

$$\begin{aligned}
\|\widehat{M} - M^{\sim w}\|_1 &= \sum_{i,j=1}^{\ell} |\widehat{M}(i,j) - M^{\sim w}(i,j)| && \text{(def. of } \|\cdot\|_1\text{)} \\
&= \sum_{i,j=1}^{\ell} |\widehat{M}(i,j) - N(i,j)| && \text{(first part of lemma)} \\
&= \sum_{i,j=1}^{\ell} \left| \sum_{a,b=1}^{k} \widehat{A}(a,b,i,j) - \sum_{a,b=1}^{k} A^{\sim w}(a,b,i,j) \right| && \text{(defs. of } \widehat{M}, N\text{)} \\
&\leq \sum_{a,b=1}^{k} \sum_{i,j=1}^{\ell} \left| \widehat{A}(a,b,i,j) - A^{\sim w}(a,b,i,j) \right| && \text{(triangle ineq.)} \\
&= \|\widehat{A} - A^{\sim w}\|_1. && \text{(def. of } \|\cdot\|_1\text{)}
\end{aligned}$$

□

6.3 Accounting for errors: Proving Corollary 3.4 and Lemma 3.22

Proof of Corollary 3.4. First, we observe that WLOG $\mathbf{t} = \mathbf{t}_{\mathrm{FJ}}$ and $\mathbf{r} = \mathbf{r}_{\mathrm{FJ}}$, i.e., there is no refinement. This is because given a general refinement $\mathbf{t} \in \mathbb{T}^{\ell}$ of $\mathbf{t}_{\mathrm{FJ}}$, and an estimate $\widehat{M} \in \mathbb{M}^{\ell}$ for $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$, we can naturally define a (less refined) estimate $\widehat{M}' \in \mathbb{M}^{\ell}_{\mathrm{FJ}}$ for $\mathsf{BiasMat}_{G,\mathbf{t}_{\mathrm{FJ}}}$ by collapsing $\mathbf{t}$-intervals in the same $\mathbf{t}_{\mathrm{FJ}}$-interval into a single entry.[16] By the triangle inequality,

[16]Formally, $\widehat{M}'$ is defined via $\widehat{M}'(i,j) = \sum_{i' : \mathsf{ind}^{\mathbf{t}_{\mathrm{FJ}}}(t_{i'}) = i} \sum_{j' : \mathsf{ind}^{\mathbf{t}_{\mathrm{FJ}}}(t_{j'}) = j} \widehat{M}(i',j')$; each entry of $\widehat{M}$ contributes to a unique entry of $\widehat{M}'$.

$\|\widehat{M}' - \mathsf{BiasMat}_{G,\mathbf{t}_{\mathrm{FJ}}}\|_1 \leq \|\widehat{M} - \mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}\|_1$. Finally, since $\mathbf{r} = (r_1, \dots, r_\ell)$ is inherited from $\mathbf{r}_{\mathrm{FJ}} = (r_1^*, \dots, r_{\ell_{\mathrm{FJ}}}^*)$, we have $\sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) = \sum_{i,j=1}^{\ell_{\mathrm{FJ}}} r_i^*(1-r_j^*)\widehat{M}'(i,j)$.

Now, let $M = \mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ and $\delta(i,j) = \widehat{M}(i,j) - M(i,j)$. We have

$$\sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) = \sum_{i,j=1}^{\ell} r_i(1-r_j)M(i,j) + \sum_{i,j=1}^{\ell} r_i(1-r_j)\delta(i,j).$$

Our goal is essentially to bound the second term on the right-hand side. Indeed, since $\|\widehat{M} - M\|_1 = \sum_{i,j=1}^{\ell} |\delta(i,j)| \leq \epsilon$, the triangle inequality gives $\left|\sum_{i,j=1}^{\ell} \delta(i,j)\right| \leq \epsilon$. Further, $0 \leq r_i, r_j \leq 1$, so $0 \leq r_i(1-r_j) \leq 1$, so $\left|\sum_{i,j=1}^{\ell} r_i(1-r_j)\delta(i,j)\right| \leq \epsilon$. Thus, we can conclude

$$\sum_{i,j=1}^{\ell} r_i(1-r_j)M(i,j) - 2\epsilon \leq \sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) - \epsilon \leq \sum_{i,j=1}^{\ell} r_i(1-r_j)M(i,j),$$

which yields the desired inequality by Lemma 3.3. □

Proof of Lemma 3.22. Recall $A = \mathsf{BiasDegArr}_{\mathcal{G},\mathbf{d},\mathbf{t}}$. By Corollary 3.21, we have $\|\widehat{A} - A^{\sim w}\|_1 \leq \epsilon_{\mathrm{err}}(k\ell)^2 + C_{\mathrm{win}}/w$. Letting $M = \mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$, by Fact 3.9 we have $M = \mathsf{Proj}(A)$. Thus, by Proposition 3.16, we have $\|\widehat{M} - M^{\sim w}\|_1 \leq \delta_{\mathrm{err}} \overset{\mathrm{def}}{=} \epsilon_{\mathrm{err}}(k\ell)^2 + C_{\mathrm{win}}/w$.

By Lemma 3.13, there exists a weighted graph $\mathcal{H}$ with bias matrix $N = \mathsf{BiasMat}_{\mathcal{H},\mathbf{t}}$ such that $\|N - M^{\sim w}\|_1 \leq \delta_{\mathrm{bias}} \overset{\mathrm{def}}{=} C_{\mathrm{smooth}}\epsilon_{\mathrm{bias}}(w+1)$ and $|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{H}}| \leq \delta_{\mathrm{bias}}$. By the triangle inequality, $\|N - \widehat{M}\|_1 \leq \delta_{\mathrm{err}} + \delta_{\mathrm{bias}}$. Now by Corollary 3.4,

$$\alpha_{\mathrm{FJ}}\mathsf{val}_{\mathcal{H}} - 2(\delta_{\mathrm{err}} + \delta_{\mathrm{bias}}) \leq \sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) - (\delta_{\mathrm{err}} + \delta_{\mathrm{bias}}) \leq \mathsf{val}_{\mathcal{H}}.$$

By our assumed bound on $\mathsf{val}_{\mathcal{H}}$, we get

$$\alpha_{\mathrm{FJ}}(\mathsf{val}_{\mathcal{G}} - \delta_{\mathrm{bias}}) - 2(\delta_{\mathrm{err}} + \delta_{\mathrm{bias}}) \leq \sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) - (\delta_{\mathrm{err}} + \delta_{\mathrm{bias}}) \leq \mathsf{val}_{\mathcal{G}} + \delta_{\mathrm{bias}},$$

and thus

$$\alpha_{\mathrm{FJ}}(\mathsf{val}_{\mathcal{G}} - \delta_{\mathrm{bias}}) - (2\delta_{\mathrm{err}} + 3\delta_{\mathrm{bias}}) \leq \sum_{i,j=1}^{\ell} r_i(1-r_j)\widehat{M}(i,j) - \delta \leq \mathsf{val}_{\mathcal{G}},$$

and the left-hand side is at least $(\alpha_{\mathrm{FJ}} - \epsilon)\mathsf{val}_{\mathcal{G}}$ since $\mathsf{val}_{\mathcal{G}} \geq \frac{1}{4}$ and $\alpha_{\mathrm{FJ}} \leq 1$. □

6.4 "Sandwiching": Proving Lemma 3.20

Now we prove Lemma 3.20, bounding the 1-norm between A^{-w} and A^{+w} for an array $A \in \mathbb{A}_{\Delta}^{k,\ell}$. We begin with the following lemma:

Lemma 6.6 ("Borders of 4D windows are effectively 3D"). *There exists a universal constant $C'_{\mathrm{win}} > 0$ such that for every $w < k, \ell \in \mathbb{N}$, and $a, b \in [k], i, j \in [\ell]$, $|\mathsf{Win}^{w+1,k,\ell}(a,b,i,j) \setminus \mathsf{Win}^{w-1,k,\ell}(a,b,i,j)| \leq C'_{\mathrm{win}} w^3$.*

Proof. Let $w_1^- = |\mathsf{Win}^{w-1,k}(a)|$ and $w_1^+ = |\mathsf{Win}^{w+1,k}(a)|$ denote the sizes of the 1-dimensional windows around a of sizes $w-1$ and $w+1$, respectively. By Fact 6.1, we have $w_1^- \le 2(w-1)+1 = 2w-1$. By Fact 6.2, we have $w_1^+ \le w_1^- + 4$.

We can similarly define $w_2^-, w_2^+, w_3^-, w_3^+, w_4^-, w_4^+$, and then we have

$$\begin{aligned}|\mathsf{Win}^{w+1,k,\ell}(a,b,i,j) - \mathsf{Win}^{w-1,k,\ell}(a,b,i,j)| &= w_1^+w_2^+w_3^+w_4^+ - w_1^-w_2^-w_3^-w_4^- \\ &\le (w_1^-+4)(w_2^-+4)(w_3^-+4)(w_4^-+4) - w_1^-w_2^-w_3^-w_4^- \\ &= 4C_3 + 16C_2 + 64C_1 + 256\end{aligned}$$

where C_t denotes the degree-t symmetric polynomial evaluated on $w_1^-, w_2^-, w_3^-, w_4^-$ (i.e., it sums the products of sets of t values). Combining with the upper-bounds on w_i^-, we conclude

$$4\binom{4}{3}(2w-1)^3 + 16\binom{4}{2}(2w-1)^2 + 64\binom{4}{1}(2w-1) + 256 = 16(8w^3+12w^2+14+5),$$

so setting $C'_{\text{win}} = 16(8+12+14+5) = 624$ is sufficient. □

Now, we prove Lemma 3.20:

Proof of Lemma 3.20. We begin by examining the difference $\delta(a,b,i,j) = A^{+w}(a,b,i,j) - A^{-w}(a,b,i,j)$ in a single entry. Note that $\delta(a,b,i,j) \ge 0$ for all a,b,i,j since A's entries are nonnegative. We have

$$\begin{aligned}\delta(a,b,i,j) = &\sum_{(a',b',i',j')\in\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)} (\nu^{+w,k,\ell}(a',b',i',j') - \nu^{-w,k,\ell}(a',b',i',j'))A(a',b',i',j') \\ &+ \sum_{(a',b',i',j')\in\mathsf{Win}^{w+1,k,\ell}(a,b,i,j)\setminus\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)} \nu^{+w,k,\ell}(a',b',i',j')A(a',b',i',j').\end{aligned}$$

In other words, the error comes from two places: Different normalizations inside $\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)$, and then extra entries in $\mathsf{Win}^{w+1,k,\ell}(a,b,i,j)$ which are not counted at all in $\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)$.

Recall that

$$\nu^{+w,k,\ell}(a',b',i',j') = \max_{(a'',b'',i'',j'')\in\mathsf{Win}^{1,k,\ell}(a',b',i',j')} 1/|\mathsf{Win}^{w,k,\ell}(a'',b'',i'',j'')|.$$

Now by the triangle inequality for windows (Fact 6.5), we have $\mathsf{Win}^{w-1,k,\ell}(a',b',i',j') \subseteq \mathsf{Win}^{w,k,\ell}(a'',b'',i'',j'')$ where (a'',b'',i'',j'') is the maximizing index in $\nu^{+w,k,\ell}$; thus, $\nu^{+w,k,\ell}(a',b',i',j') \le 1/|\mathsf{Win}^{w-1,k,\ell}(a',b',i',j')|$. Similarly, $\nu^{-w,k,\ell}(a',b',i',j') \ge 1/|\mathsf{Win}^{w+1,k,\ell}(a',b',i',j')|$. So

$$\nu^{+w,k,\ell}(a',b',i',j') - \nu^{-w,k,\ell}(a',b',i',j') \le \frac{|\mathsf{Win}^{w+1,k,\ell}(a',b',i',j')| - |\mathsf{Win}^{w-1,k,\ell}(a',b',i',j')|}{|\mathsf{Win}^{w+1,k,\ell}(a',b',i',j')|\cdot|\mathsf{Win}^{w-1,k,\ell}(a',b',i',j')|} \le C'_{\text{win}}w^{-5},$$

where we upper-bound the numerator by $C'_{\text{win}}w^3$ with Lemma 6.6 and lower-bound the denominator by w^8 with Fact 6.1. Hence we can write

$$\begin{aligned}\delta(a,b,i,j) \le C'_{\text{win}}w^{-5} &\sum_{(a',b',i',j')\in\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)} A(a',b',i',j') \\ + w^{-4} &\sum_{(a',b',i',j')\in\mathsf{Win}^{w+1,k,\ell}(a,b,i,j)\setminus\mathsf{Win}^{w-1,k,\ell}(a,b,i,j)} A(a',b',i',j'),\end{aligned}$$

where for the second term we use that $\nu^{+w,k,\ell}(a',b',i',j') \geq 1/|\mathsf{Win}^{w+1,k,\ell}(a',b',i',j')| \geq w^{-4}$ (by Facts 6.1 and 6.5).

Finally, we bound $\|A^{+w} - A^{-w}\|_1 = \sum_{a,b=1}^{k} \sum_{i,j=1}^{\ell} \delta(a,b,i,j)$ by double-counting and symmetry of inclusion in windows (Facts 6.3 and 6.4). In particular, an entry $A(a',b',i',j')$ will be counted $|\mathsf{Win}^{w-1,k,\ell}(a',b',i',j')| \leq 16w^4$ times in the first sum (by Fact 6.1), and $|\mathsf{Win}^{w+1,k,\ell}(a',b',i',j') \setminus \mathsf{Win}^{w-1,k,\ell}(a',b',i',j')| \leq C'_{\text{win}} w^3$ times in the second sum, giving an overall bound of $\|A^{+w} - A^{-w}\|_1 \leq 17\|A\|_1 C'_{\text{win}}/w$, which is the desired bound when $C_{\text{win}} = 17C'_{\text{win}}$ (since $\|A\|_1 = 1$ by assumption). $\square$

6.5 "Smoothing graphs": Proving Lemma 3.13

In this section, we prove Lemma 3.13. We construct $\mathcal{H}$ based on $\mathcal{G}$ in two stages: First, we construct an intermediate (weighted) graph $\mathcal{K}$ by "blowing up" each vertex of $\mathcal{G}$ (which preserves biases of vertices and the Max-DICUT value). Then, we slightly modify $\mathcal{K}$ to form the (weighted) graph $\mathcal{H}$ by perturbing the bias at each vertex, and argue that this does not change the Max-DICUT value too much.

We begin with some notation. We label the vertices of $\mathcal{G}$ by $[n] = \{1,\ldots,n\}$. We assume WLOG that $\mathcal{G}$ has no isolated vertices (if not, we simply remove all isolated vertices and analyze the new graph; this doesn't affect $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ or $\mathsf{val}_{\mathcal{G}}$).

6.5.1 Defining $\mathcal{K}$ via (weighted) blowups

First, given $\mathcal{G}$, we construct a graph $\mathcal{K}$ as follows. For each vertex $v \in [n]$, in $\mathcal{K}$ we create $|\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|$ distinct vertices labeled $\{v\} \times \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))$. Thus, $\mathcal{K}$ has vertex-set $\mathcal{V} \overset{\text{def}}{=} \bigcup_{v=1}^{n} \{v\} \times \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))$.

Now if $\mathsf{AdjMat}_{\mathcal{G}}(u,v)$ is the (positive) weight of an edge $u \to v$ in $\mathcal{G}$, we divide its weight evenly among all the copies of u and v in $\mathcal{K}$, i.e., we place weight

$$\mathsf{AdjMat}_{\mathcal{K}}(u,i',v,j') \overset{\text{def}}{=} \nu^{\sim w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)) \cdot \mathsf{AdjMat}_{\mathcal{H}}(u,v) \tag{6.7}$$

on all edges $(u,i') \to (v,j')$ for $(i',j') \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))$.[17]

We claim that $\mathcal{K}$ has the following properties:

Claim 6.8. *For every vertex $v \in [n]$ and $i' \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))$, we have:*

1. $\mathsf{deg\text{-}out}_{\mathcal{K}}(v,i') = \mathsf{deg\text{-}out}_{\mathcal{G}}(v) \cdot |\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|^{-1}$.
2. $\mathsf{deg\text{-}in}_{\mathcal{K}}(v,i') = \mathsf{deg\text{-}in}_{\mathcal{G}}(v) \cdot |\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|^{-1}$.
3. $\mathsf{deg}_{\mathcal{K}}(v,i') = \mathsf{deg}_{\mathcal{G}}(v) \cdot |\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|^{-1}$.
4. $\mathsf{bias}_{\mathcal{K}}(v,i') = \mathsf{bias}_{\mathcal{G}}(v)$.

Further, $m_{\mathcal{G}} = m_{\mathcal{K}}$ and $\mathsf{val}_{\mathcal{G}} = \mathsf{val}_{\mathcal{K}}$.

The proof is basically a series of double-counting arguments using the definition of $\mathcal{K}$, and we defer it until the end of the section.

[17] Recall $\nu^{\sim w,\ell}(i,j) = |\mathsf{Win}^{w,\ell}(i,j)|$.

6.5.2 Constructing $\mathcal{H}$ via perturbations

Now, we construct $\mathcal{H}$ by modifying $\mathcal{K}$. We begin with a new graph $\mathcal{H}_0$ consisting of a copy of $\mathcal{K}$ and a new isolated vertex $(\star,\star)$, so that $\mathcal{H}_0$ has vertex-set $\mathcal{V} \cup \{(\star,\star)\}$, and let $\mathcal{V}^* = \bigcup_{v\in[n]}(\{v\} \times (\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))) \setminus \{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v)\})) \subseteq \mathcal{V}$ denote the set of vertices whose biases we'd like to modify. We let $m_0 \overset{\text{def}}{=} m_{\mathcal{G}} = m_{\mathcal{K}} = m_{\mathcal{H}_0}$ denote the total weight in $\mathcal{G}$, $\mathcal{K}$, and $\mathcal{H}_0$ (which coincide by Claim 6.8).

Next, we iteratively modify the biases of vertices using the following claim, which roughly lets us decrease the bias of a vertex $(v,i') \in \mathcal{V}^*$ by increasing its in-degree and decreasing its out-degree, or increase the bias of a vertex by increasing its out-degree and decreasing its in-degree, "without modifying the graph too much".

Claim 6.9. *Let $\mathcal{L}$ be any weighted graph on vertex-set $\mathcal{V} \cup \{(\star,\star)\}$. For every $(v,i') \in \mathcal{V}^*$, $0 \le \alpha$, and $0 \le \beta \le \mathsf{deg\text{-}out}_{\mathcal{L}}(v,i')$, there exists an "updated" graph $\mathcal{L}'$ on vertex-set $\mathcal{V} \cup \{(\star,\star)\}$, such that:*

1. *"All vertices except (v,i') and $(\star,\star)$ stay the same": For all $(u,j') \neq (v,i') \in \mathcal{V}$, $\mathsf{deg\text{-}out}_{\mathcal{L}'}(u,j') = \mathsf{deg\text{-}out}_{\mathcal{L}}(u,j')$ and $\mathsf{deg\text{-}in}_{\mathcal{L}'}(u,j') = \mathsf{deg\text{-}in}_{\mathcal{L}}(u,j')$.*
2. *"(v,i')'s change is controlled": $\mathsf{deg\text{-}out}_{\mathcal{L}'}(v,i') = \mathsf{deg\text{-}out}_{\mathcal{L}}(v,i') - \beta$ and $\mathsf{deg\text{-}in}_{\mathcal{L}'}(v,i') = \mathsf{deg\text{-}in}_{\mathcal{L}}(v,i') + \alpha$.*
3. *"The new weight added is bounded": $m_{\mathcal{L}'} = m_{\mathcal{L}} + \alpha$.*
4. *"Changes in weight are bounded": $\|\mathsf{AdjMat}_{\mathcal{L}'} - \mathsf{AdjMat}_{\mathcal{L}}\|_1 = \alpha + 2\beta$.*

The same holds when we swap $\mathsf{deg\text{-}out}$*'s with* $\mathsf{deg\text{-}in}$*'s.*

A few quick remarks: Note that $\mathsf{AdjMat}_{\mathcal{L}}$ and $\mathsf{AdjMat}_{\mathcal{L}'}$ are *not* normalized, so in Item 4 we are simply summing the magnitude of the change in weight from $\mathcal{L}$ to $\mathcal{L}'$ over every edge. Also, in general the out- or in-degree of $(\star,\star)$ will have to change in order to ensure the first two items hold; the second two items state that this doesn't cause too many problems for us.

Proof of Claim 6.9. Suppose we are decreasing the bias of (v,i') — the other case is analogous. Roughly, here is the process: We start by setting $\mathcal{L}'$ to be the same as $\mathcal{L}$; we will only adjust the weights of a few edges. We increase the weight of the edge $(\star,\star) \to (v,i')$ by α. Then, we arbitrarily distribute weight β among out-edges $(v,i') \to (u,j')$ and then transfer this distribution to edges $(\star,\star) \to (u,j')$.

More formally, we do the following. For each $(u,j') \neq (v,i') \in \mathcal{V}$, we pick a coefficient $0 \le \beta_{(u,j')} \le \mathsf{AdjMat}_{\mathcal{L}}(v,i',u,j')$ such that $\sum_{(u,j')\neq(v,i')\in\mathcal{V}} \beta_{(u,j')} = \beta$; this is possible as $\beta \le \mathsf{deg\text{-}out}_{\mathcal{L}}(v,i') = \sum_{(u,j')\neq(v,i')\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{L}}(v,i',u,j')$ by assumption, and so $\beta_{(u,j')}$'s can be picked greedily. Now we define the adjacency matrix of the new graph $\mathcal{L}'$ as follows: We set $\mathsf{AdjMat}_{\mathcal{L}'}(\star,\star,v,i') = \mathsf{AdjMat}_{\mathcal{L}}(\star,\star,v,i') + \alpha$. For each $(u,j') \neq (v,i') \in \mathcal{V}$, we set $\mathsf{AdjMat}_{\mathcal{L}'}(v,i',u,j') = \mathsf{AdjMat}_{\mathcal{L}}(v,i',u,j') - \beta_{(u,j')}$ and $\mathsf{AdjMat}_{\mathcal{L}'}(\star,\star,u,j') = m_{\mathcal{L}}(\star,\star,u,j') + \beta_{(u,j')}$. Finally, for all remaining edges, i.e., $(w,k'),(u,j') \neq (v,i') \in \mathcal{V}$, we set $\mathsf{AdjMat}_{\mathcal{L}'}(w,k',u,j') = \mathsf{AdjMat}_{\mathcal{L}}(w,k',u,j')$. The four desiderata follow immediately from this construction. □

Now consider an arbitrary vertex $(v,i') \in \mathcal{V}^*$, and suppose WLOG $i' < \mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v)$, so we are aiming to decrease $\mathsf{bias}_{\mathcal{H}_0}(v,i')$. Arbitrarily pick some "target bias" b^* such that $\mathsf{ind}^{\mathbf{t}}(b^*) = i'$. Consider setting

$$\alpha, \beta := \frac{1}{2}(\mathsf{bias}_{\mathcal{H}_0}(v,i') - b^*)\, \mathsf{deg}_{\mathcal{H}_0}(v,i').$$

Since $\mathsf{bias}_{\mathcal{H}_0}(v,i') = \mathsf{bias}_{\mathcal{G}}(v)$ and $i' \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{H}_0}(v))$, we know $(\mathsf{bias}_{\mathcal{H}_0}(v,i') - b^*) \le \epsilon_{\mathrm{bias}}(w+1)$. Then since $\mathbf{t}$ is ϵ_{bias}-wide, we have

$$\alpha, \beta \le \frac{1}{2}\epsilon_{\mathrm{bias}}(w+1)\,\mathsf{deg}_{\mathcal{H}_0}(v,i'). \tag{6.10}$$

Moreover, if we apply Claim 6.9 with $\mathcal{L} = \mathcal{H}_0$, then we get a new graph $\mathcal{H}_1 = \mathcal{L}'$ in which (v,i') has bias

$$\begin{aligned}\mathsf{bias}_{\mathcal{H}_1}(v,i') &= \frac{(\mathsf{deg\text{-}out}_{\mathcal{H}_0}(v,i') - \beta) - (\mathsf{deg\text{-}in}_{\mathcal{H}_0}(v,i') + \alpha)}{(\mathsf{deg\text{-}out}_{\mathcal{H}_0}(v,i') - \beta) + (\mathsf{deg\text{-}in}_{\mathcal{H}_0}(v,i') + \alpha)} \\ &= \mathsf{bias}_{\mathcal{H}_0}(v,i') - \frac{2\beta}{\mathsf{deg}_{\mathcal{H}_0}(v,i')} \\ &= b^*\end{aligned}$$

and the in- and out-degrees of all other vertices except $(\star,\star)$ are fixed. We can apply this procedure to all vertices in $\mathcal{V}^*$ whose biases need to be decreased, along with the analogous procedure for vertices in $\mathcal{V}^*$ whose biases need to be *increased*. After applying this operation to every vertex in $\mathcal{V}^*$, we end up with a graph which we'll denote $\mathcal{H}$, which has the following properties:

Claim 6.11. *For* every *vertex* $(v,i') \in \mathcal{V}$, $\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{H}}(v,i') = i'$. *Also,* $m_0 \le m_{\mathcal{H}} \le (1+\epsilon_{\mathrm{bias}}(w+1))m_0$ *and* $\|\mathsf{AdjMat}_{\mathcal{H}} - \mathsf{AdjMat}_{\mathcal{H}_0}\|_1 \le 3\epsilon_{\mathrm{bias}}(w+1)m_0$.

Proof. When we constructed $\mathcal{H}$ from $\mathcal{H}_0$, when modifying a vertex $(v,i') \in \mathcal{V}^*$, Eq. (6.10) says we set $\alpha,\beta \le \frac{1}{2}\epsilon_{\mathrm{bias}}(w+1)\mathsf{deg}_{\mathcal{H}_0}(v,i')$; hence, by Claim 6.9,

$$m_0 \le m_{\mathcal{H}} \le m_0 + \frac{1}{2}\sum_{(v,i')\in\mathcal{V}^*} \epsilon_{\mathrm{bias}}(w+1)\mathsf{deg}_{\mathcal{H}_0}(v,i') = (1+\epsilon_{\mathrm{bias}}(w+1))m_0$$

since $m_0 = m_{\mathcal{H}} = \frac{1}{2}\sum_{(v,i')\in\mathcal{V}\cup\{(\star,\star)\}} \mathsf{deg}_{\mathcal{H}}(v,i'))$ is the total weight in $\mathcal{H}$ and $\mathcal{V}^* \subseteq \mathcal{V}\cup\{(\star,\star)\}$. Similarly, by Claim 6.9 and the triangle inequality, we can bound the distance between the *unnormalized* adjacency matrices of $\mathcal{H}$ and $\mathcal{H}_0$:

$$\|\mathsf{AdjMat}_{\mathcal{H}} - \mathsf{AdjMat}_{\mathcal{H}_0}\|_1 \le \frac{1}{2}\sum_{(v,i')\in\mathcal{V}^*} 3\epsilon_{\mathrm{bias}}(w+1)\mathsf{deg}_{\mathcal{H}_0}(v,i') \le 3\epsilon_{\mathrm{bias}}(w+1)m_0.$$

□

6.5.3 Proving that $\mathcal{H}$ fulfills the desiderata

Now, we claim that $\mathcal{H}$ fulfills the two desiderata (i.e., the upper bounds on $|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{H}}|$ and $\|\widetilde{\mathsf{BiasMat}}^w_{\mathcal{G},\mathbf{t}} - \mathsf{BiasMat}_{\mathcal{H},\mathbf{t}}\|_1$), proving the lemma.

Proof of Lemma 3.13. According to our definition, both $\mathsf{val}_{\mathcal{G}}$ and $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$ are normalized by $m_{\mathcal{G}}$ (the total weight in $\mathcal{G}$), and the same goes for $\mathcal{H}$. Towards both proofs, it will be convenient for us to first bound the distance between the *unnormalized* variants of these objects, and then argue that $m_{\mathcal{H}} \approx m_{\mathcal{G}}$ and therefore the normalized versions are close as well. We set $C_{\mathrm{smooth}} = 7$.

Upper-bounding $|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{H}}|$. Recall that $\mathsf{val}_{\mathcal{K}} = \mathsf{val}_{\mathcal{G}}$. Further, $\mathsf{val}_{\mathcal{H}_0} = \mathsf{val}_{\mathcal{K}}$ as $\mathcal{H}_0$ simply adds an isolated vertex (i.e., $(\star,\star)$).

Now let

$$h = m_{\mathcal{H}}\mathsf{val}_{\mathcal{H}} \text{ and } g = m_0 \mathsf{val}_{\mathcal{H}_0}$$

denote the *unnormalized* $\mathsf{Max\text{-}DICUT}$ values of $\mathcal{H}$ and $\mathcal{H}_0$, respectively. We claim that $|g-h| \le \|\mathsf{AdjMat}_{\mathcal{H}} - \mathsf{AdjMat}_{\mathcal{H}_0}\|_1$ (and thus, $|g-h| \le 3\epsilon_{\text{bias}}(w+1)m_0$ by Claim 6.11). Observe that

$$|g-h| = \left| \max_{\mathbf{y}\in\{0,1\}^{\mathcal{V}^*}} m_{\mathcal{H}}\mathsf{val}_{\mathcal{H}}(\mathbf{y}) - \max_{\mathbf{y}\in\{0,1\}^{\mathcal{V}^*}} m_0 \mathsf{val}_{\mathcal{H}_0}(\mathbf{y}) \right| \le \left| \max_{\mathbf{y}\in\{0,1\}^{\mathcal{V}^*}} (m_{\mathcal{H}}\mathsf{val}_{\mathcal{H}}(\mathbf{y}) - m_0\mathsf{val}_{\mathcal{H}_0}(\mathbf{y})) \right| .$$

Indeed, if $\mathbf{y} \in \{0,1\}^{\mathcal{V}^*}$ is any assignment, we have

$$\begin{aligned}
|m_{\mathcal{H}}\mathsf{val}_{\mathcal{H}}(\mathbf{y}) - m_0\mathsf{val}_{\mathcal{H}_0}(\mathbf{y})| &= \left| \sum_{(u,i'),(v,j')\in\mathcal{V}\cup\{(\star,\star)\}} (\mathsf{AdjMat}_{\mathcal{H}}(u,i',v,j') - \mathsf{AdjMat}_{\mathcal{H}_0}(u,i',v,j')) \cdot y_{(u,i')}(1-y_{(v,j')}) \right| \\
&\le \sum_{(u,i'),(v,j')\in\mathcal{V}\cup\{(\star,\star)\}} \left| \mathsf{AdjMat}_{\mathcal{H}}(u,i',v,j') - \mathsf{AdjMat}_{\mathcal{H}_0}(u,i',v,j') \right| \\
&\qquad \text{(tri. ineq. and } y_{(u,i')} \in \{0,1\}) \\
&= \|\mathsf{AdjMat}_{\mathcal{H}} - \mathsf{AdjMat}_{\mathcal{H}_0}\|_1. \qquad \text{(def. of } \|\cdot\|_1)
\end{aligned}$$

Finally, we can write

$$|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{H}}| = \left| \frac{1}{m_0} g - \frac{1}{m_{\mathcal{H}}} h \right| = \left| \frac{1}{m_0}(g-h) + \eta \right| \le 3\epsilon_{\text{bias}}(w+1) + \eta$$

by the triangle inequality, where $\eta = h(1/m_0 - 1/m_{\mathcal{H}}) = h(m_{\mathcal{H}} - m_0)/(m_{\mathcal{H}}m_0)$ is a "normalization error" term. We can upper bound this error term:

$$\eta = \frac{m_{\mathcal{H}}(m_{\mathcal{H}} - m_0)\mathsf{val}_{\mathcal{H}}}{m_{\mathcal{H}}m_0} \le \frac{m_{\mathcal{H}} - m_0}{m_{\mathcal{H}}} \le \epsilon_{\text{bias}}(w+1)$$

using $\mathsf{val}_{\mathcal{H}} \le 1$ and Claim 6.11.

Upper-bounding $\|\mathsf{BiasMat}^{\sim w}_{\mathcal{G},\mathbf{t}} - \mathsf{BiasMat}_{\mathcal{H},\mathbf{t}}\|_1$. Now, we analyze the distance between the unnormalized matrices $G = m_0\mathsf{BiasMat}^{\sim w}_{\mathcal{G},\mathbf{t}}$ and $H = m_{\mathcal{H}}\mathsf{BiasMat}_{\mathcal{H},\mathbf{t}}$. This suffices, similarly to the case of the $\mathsf{Max\text{-}DICUT}$ values, as by the triangle inequality

$$\left\| \frac{1}{m_0} G - \frac{1}{m_{\mathcal{H}}} H \right\|_1 \le \frac{1}{m_0}\|G - H\|_1 + \eta$$

where $\eta = \|H\|_1(1/m_0 - 1/m_{\mathcal{H}}) \le \epsilon_{\text{bias}}(w+1)$ as $\|H\|_1 = m_{\mathcal{H}}$.

The basic idea here is that if we consider an edge (u,v) in $\mathcal{G}$, its weight $\mathsf{AdjMat}_{\mathcal{G}}(u,v)$ will contribute (up to normalization) to the entries corresponding to $\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))$ in $\mathsf{BiasMat}^{\sim w}_{\mathcal{G},\mathbf{t}}$. In particular, for each position $(i',j') \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))$ which (u,v) contributes to in $\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}$, we have created a copy (u,i',v,j') in $\mathcal{H}$ which will contribute to the corresponding entry in $\mathsf{BiasMat}_{\mathcal{H},\mathbf{t}}$! This correspondence between contributions is not exact: our construction of $\mathcal{H}$ slightly modified the weights inherited from $\mathcal{G}$, and introduced a new vertex $(\star,\star)$ with uncontrolled bias, but these issues can be taken care of with our bound on $\|\mathsf{AdjMat}_{\mathcal{H}} - \mathsf{AdjMat}_{\mathcal{H}_0}\|_1$.

By definition, we have

$$\begin{aligned}H(i,j) = &\sum_{(u,i'),(v,j')\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{H}}(u,i',v,j')\mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{H}}(u,i',v,j')=(i,j)} \\ &+ \sum_{(u,i')\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{H}}(u,i',\star,\star)\mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{H}}(u,i',\star,\star)=(i,j)} \\ &+ \sum_{(v,j')\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{H}}(\star,\star,v,j')\mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{H}}(\star,\star,v,j')=(i,j)} \\ &+ \mathsf{AdjMat}_{\mathcal{H}}(\star,\star,\star,\star)\mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{H}}(\star,\star,\star,\star)=(i,j)}.\end{aligned}$$

Now define the matrix H' by "forgetting about $(\star,\star)$", i.e., we only take the first term in the expansion of H:

$$H'(i,j) = \sum_{(u,i'),(v,j')\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{H}}(u,i',v,j')\mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{H}}(u,i',v,j')=(i,j)}.$$

Note that $\|H-H'\|_1 = \mathsf{deg}_{\mathcal{H}}(\star,\star)$ since every edge in $\mathcal{H}$ contributes to precisely one entry. Further, since $(\star,\star)$ was isolated in $\mathcal{H}_0$, Claims 6.9 and 6.11 imply $\mathsf{deg}_{\mathcal{H}}(\star,\star) \le 3\epsilon_{\text{bias}}(w+1)m_0$. Thus, by the triangle inequality, it suffices to prove that $\|G-H'\|_1 \le 3\epsilon_{\text{bias}}(w+1)m_0$.

Now H''s entries sum only over vertices in $\mathcal{V}$ (i.e., not $(\star,\star)$), and we constructed $\mathcal{H}$ so that for each vertex (v,i') we have $\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{H}}(v,i') = i'$ (Claim 6.11). Thus, the (i,j)-th entry in H' counts only edges (u,i',v,j') such that $i=i'$ and $j=j'$, so we can write

$$H'(i,j) = \sum_{(u,i)\in\mathcal{V}}\sum_{(v,j)\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{H}}(u,i,v,j).$$

On the other hand, we have

$$m_0\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}(i,j) = \sum_{u,v=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v)\mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)=(i,j)}$$

and so we have

$$\begin{aligned}G(i,j) &= m_0 \sum_{(i',j')\in\mathsf{Win}^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j')\mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}(i',j') && \text{(def. of smoothing)} \\ &= m_0 \sum_{(i',j')\in\mathsf{Win}^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j')\cdot\frac{1}{m_0}\sum_{u,v=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v)\mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)=(i',j')} \\ & && \text{(def. of } \mathsf{BiasMat}_{\mathcal{G},\mathbf{t}}) \\ &= \sum_{(i',j')\in\mathsf{Win}^{w,\ell}(i,j)} \nu^{\sim w,\ell}(i',j')\sum_{u,v=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v)\mathbb{1}_{\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)=(i',j')}.\end{aligned}$$

Now an edge (u,v) has a nonzero contribution iff $(i,j) \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))$, i.e., if $(u,i),(v,j)\in\mathcal{V}$; in this case, its contribution is precisely $\nu^{\sim w,\ell}((\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)))\cdot\mathsf{AdjMat}_{\mathcal{G}}(u,v) = \mathsf{AdjMat}_{\mathcal{K}}(u,i,v,j) = \mathsf{AdjMat}_{\mathcal{H}_0}(u,i,v,j)$, so we can write

$$G(i,j) = \sum_{(v,i)\in\mathcal{V}}\sum_{(u,j)\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{H}_0}(u,i,v,j).$$

This matches our expression for $H'(i,j)$, except that the weights come from $\mathcal{H}_0$ instead of $\mathcal{H}$. To complete the analysis:

$$\begin{aligned}
\|G - H'\|_1 &= \sum_{i,j=1}^{\ell} \left| \sum_{(u,i)\in\mathcal{V}} \sum_{(v,j)\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{H}_0}(u,i,v,j) - \sum_{(u,i)\in\mathcal{V}} \sum_{(v,j)\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{H}}(u,i,v,j) \right| \\
&\le \sum_{i,j=1}^{\ell} \sum_{(u,i)\in\mathcal{V}} \sum_{(v,j)\in\mathcal{V}} \left| \mathsf{AdjMat}_{\mathcal{H}_0}(u,i,v,j) - \mathsf{AdjMat}_{\mathcal{H}}(u,i,v,j) \right| && \text{(triangle ineq.)} \\
&\le \sum_{(u,i')\in\mathcal{V}} \sum_{(v,j')\in\mathcal{V}} \left| \mathsf{AdjMat}_{\mathcal{H}_0}(u,i',v,j') - \mathsf{AdjMat}_{\mathcal{H}}(u,i',v,j') \right| && \text{(each term occurs once)} \\
&\le \|\mathsf{AdjMat}_{\mathcal{H}} - \mathsf{AdjMat}_{\mathcal{H}_0}\|_1 \\
&\le 3\epsilon_{\text{bias}}(w+1)m_0, && \text{(Claim 6.11)}
\end{aligned}$$

as desired. □

6.5.4 Proof of Claim 6.8

Proof of Claim 6.8. For any $v \in [n]$, any particular copy (v,i') of a vertex v has out-degree

$$\begin{aligned}
\mathsf{deg\text{-}out}_{\mathcal{K}}(v,i') &= \sum_{u=1}^{n} \sum_{j'\in\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u))} \mathsf{AdjMat}_{\mathcal{K}}(v,i',u,j') && \text{(def. of } \mathsf{deg\text{-}out}) \\
&= \sum_{u=1}^{n} \sum_{j'\in\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u))} |\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))|^{-1} \cdot \mathsf{AdjMat}_{\mathcal{G}}(u,v) && \text{(def. of } \mathcal{K}) \\
&= |\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|^{-1} \sum_{u=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v) && \text{(def. of 2D windows)} \\
&= |\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|^{-1}\, \mathsf{deg\text{-}out}_{\mathcal{G}}(v). && \text{(def. of } \mathsf{deg\text{-}out})
\end{aligned}$$

Similarly, $\mathsf{deg\text{-}in}_{\mathcal{K}}(v,i') = \mathsf{deg\text{-}in}_{\mathcal{G}}(v) \cdot |\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|^{-1}$. Thus, their sum $\mathsf{deg\text{-}out}_{\mathcal{K}}(v,i') = \mathsf{deg\text{-}out}_{\mathcal{G}}(v) \cdot |\mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|^{-1}$, and their normalized difference $\mathsf{bias}_{\mathcal{K}}(v,i') = \mathsf{bias}_{\mathcal{G}}(v)$.

Next, we observe that

$$\begin{aligned}
m_{\mathcal{K}} &= \sum_{(v,i')\in\mathcal{V}} \mathsf{deg\text{-}out}_{\mathcal{K}}(v,i') \\
&= \sum_{v=1}^{n} \sum_{i'\in\mathsf{Win}^{\sim w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))} \mathsf{deg\text{-}out}_{\mathcal{G}}(v) \cdot |\mathsf{Win}^{\sim w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))|^{-1} && \text{(defs. of } \mathcal{K} \text{ and } \mathcal{V}) \\
&= \sum_{v=1}^{n} \mathsf{deg\text{-}out}_{\mathcal{G}}(v) \\
&= m_{\mathcal{G}}.
\end{aligned}$$

Now, we claim that $\mathsf{val}_{\mathcal{G}} \le \mathsf{val}_{\mathcal{H}}$. Indeed, let $\mathbf{x}$ be any assignment for $\mathcal{G}$. We can construct an assignment $\mathbf{y} \in \{0,1\}^{\mathcal{V}}$ to $\mathcal{K}$'s vertices by simply giving each copy of a vertex in $\mathcal{K}$ its assignment in $\mathcal{G}$, i.e., $y_{(v,i')} = x_v$. This assignment has value

$$\mathsf{val}_{\mathcal{K}}(\mathbf{y}) = \sum_{(u,i'),(v,j')\in\mathcal{V}} \mathsf{AdjMat}_{\mathcal{K}}(u,i',v,j') y_{(u,i')}(1 - y_{(v,j')}) \qquad \text{(def. of } \mathsf{val})$$

$$
\begin{aligned}
&= \sum_{u,v=1}^{n} \sum_{(i',j') \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))} \mathsf{AdjMat}_{\mathcal{K}}(u,i',v,j')\; y_{(u,i')}(1-y_{(v,j')}) && \text{(def. of } \mathcal{V}) \\
&= \sum_{u,v=1}^{n} \sum_{(i',j') \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))} \nu^{\sim w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)) \cdot \mathsf{AdjMat}_{\mathcal{G}}(u,v)\; y_{(u,i')}(1-y_{(v,j')}) \\
& && \text{(def. of } \mathcal{K}) \\
&= \sum_{u,v=1}^{n} \sum_{(i',j') \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))} \nu^{\sim w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)) \cdot \mathsf{AdjMat}_{\mathcal{G}}(u,v)\; x_u(1-x_v) && \text{(def. of } \mathbf{y}) \\
&= \sum_{u,v=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v)\; x_u(1-x_v) && \text{(def. of } \nu^{\sim w,\ell}) \\
&= \mathsf{val}_{\mathcal{G}}(\mathbf{x}). && \text{(def. of } \mathsf{val})
\end{aligned}
$$

Conversely, let $\mathbf{y}$ be any assignment for $\mathcal{K}$. We construct a *probabilistic* assignment $\mathbf{x}$ by, for each vertex $v \in [n]$, sampling $i' \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(v))$ uniformly and independently at random, and setting $x_v = y_{(v,i')}$. This assignment has expected value

$$
\begin{aligned}
\mathbb{E}[\mathsf{val}_{\mathcal{G}}(\mathbf{x})] &= \mathbb{E}\left[\sum_{u,v=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v) x_u(1-x_v)\right] && \text{(def. of } \mathsf{val}) \\
&= \sum_{u,v=1}^{n} \mathsf{AdjMat}_{\mathcal{G}}(u,v)\, \mathbb{E}[x_u(1-x_v)] && \text{(lin. of } \mathbb{E}) \\
&= \sum_{u=1}^{n} \sum_{v \neq u} \mathsf{AdjMat}_{\mathcal{G}}(u,v)\, \mathbb{E}[x_u(1-x_v)] && \text{(no self-loops in } \mathcal{G}) \\
&= \sum_{u=1}^{n} \sum_{v \neq u} \mathsf{AdjMat}_{\mathcal{G}}(u,v)\, \mathbb{E}[x_u]\, \mathbb{E}[1-x_v] && \text{(independence)} \\
&= \sum_{u,v=1}^{n} \sum_{(i',j') \in \mathsf{Win}^{w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v))} \nu^{\sim w,\ell}(\mathsf{b\text{-}ind}^{\mathbf{t}}_{\mathcal{G}}(u,v)) \cdot \mathsf{AdjMat}_{\mathcal{G}}(u,v)\; y_{(u,i')}(1-y_{(v,j')}), \\
& && \text{(def. of } \mathbf{x})
\end{aligned}
$$

which equals $\mathsf{val}_{\mathcal{K}}(\mathbf{y})$ by the previous calculation. □

References

[BHP+22] Joanna Boyland, Michael Hwang, Tarun Prasad, Noah Singer, and Santhoshini Velusamy. On sketching approximations for symmetric Boolean CSPs. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2022, Virtual, September 19-21, 2022)*, volume 245 of *LIPIcs*, pages 38:1–38:23. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, July 2022.

[CGS$^+$22a] Chi-Ning Chou, Alexander Golovnev, Amirbehshad Shahrasbi, Madhu Sudan, and Santhoshini Velusamy. Sketching Approximability of (Weak) Monarchy Predicates. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2022.

[CGS$^+$22b] Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, Ameya Velingker, and Santhoshini Velusamy. Linear Space Streaming Lower Bounds for Approximating CSPs. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing*, 2022.

[CGSV21] Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all finite CSPs with linear sketches. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021, Denver, CO, USA, February 7-10, 2022)*. IEEE Computer Society, 2021.

[CGV20] Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal Streaming Approximations for all Boolean Max-2CSPs and Max-$k$SAT. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020, Virtual, November 16-19, 2020)*, pages 330–341. IEEE Computer Society, November 2020.

[CL06] Fan Chung and Linyuan Lu. Concentration Inequalities and Martingale Inequalities: A Survey. *Internet Mathematics*, 3(1):79–127, January 2006.

[FJ15] Uriel Feige and Shlomo Jozeph. Oblivious Algorithms for the Maximum Directed Cut Problem. *Algorithmica*, 71(2):409–428, February 2015.

[GKK$^+$08] Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential separation for one-way quantum communication complexity, with applications to cryptography. *SIAM Journal on Computing*, 38(5):1695–1708, December 2008.

[GT19] Venkatesan Guruswami and Runzhou Tao. Streaming Hardness of Unique Games. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2019, Cambridge, MA, USA, September 20-22, 2019)*, volume 145 of *LIPIcs*, pages 5:1–5:12. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, September 2019.

[GVV17] 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 2017, Berkeley, CA, USA, August 16-18, 2017)*, volume 81 of *LIPIcs*, pages 8:1–8:19. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, August 2017.

[HZ01] Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001, Washington, DC, USA, January 7-9, 2001)*, pages 1–7, 2001.

[Jof74] A. Joffe. On a Set of Almost Deterministic $k$-Independent Random Variables. *The Annals of Probability*, 1974.

[KK15] Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 6th Annual Conference on Innovations in Theoretical Computer Science (ITCS 2015, Rehovot, Israel, January 11-13, 2015)*, pages 367–376. Association for Computing Machinery, 2015.

[KK19] Michael Kapralov and Dmitry Krachun. An optimal space lower bound for approximating MAX-CUT. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019, Phoenix, AZ, USA, June 23-26, 2019)*, pages 277–288. Association for Computing Machinery, June 2019.

[KKS14] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014, Portland, OR, USA, January 5-7, 2014)*, pages 734–751, USA, January 2014. Society for Industrial and Applied Mathematics.

[KKS15] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015, San Diego, California, USA, January 4-6, 2015)*, pages 1263–1282. Society for Industrial and Applied Mathematics, January 2015.

[KKSV17] Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. $(1+\omega(1))$-approximation to MAX-CUT requires linear space. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017, Barcelona, Spain, January 16-19, 2017)*, pages 1703–1722. Society for Industrial and Applied Mathematics, January 2017.

[Sin22] Noah Singer. *On Streaming Approximation Algorithms for Constraint Satisfaction Problems*. Undergraduate thesis, Harvard University, Cambridge, MA, March 2022.

[SSSV23] Raghuvansh R. Saxena, Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming complexity of CSPs with randomly ordered constraints. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms*, 2023.

[SSV21] Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming approximation resistance of every ordering CSP. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2021, August 16-18, 2021)*, volume 207 of *LIPIcs*, pages 17:1–17:19. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, July 2021.

[Sud22] Madhu Sudan. Streaming and Sketching Complexity of CSPs: A survey (Invited Talk). In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming*, volume 229 of *LIPIcs*, pages 5:1–5:20. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2022.

[Tre98] Luca Trevisan. Parallel Approximation Algorithms by Positive Linear Programming. *Algorithmica*, 21(1):72–88, May 1998.

A Proofs of preliminary lemmas

Proof of Lemma 2.9. For convenience, let $m' = m_{\mathcal{G}_{\text{spar}}}$ denote the number of edges in $\mathcal{G}_{\text{spar}}$, and let $p = p_{\text{spar}}$; for an assignment $\mathbf{x} \in \{0,1\}^V$, let $y(\mathbf{x}) \overset{\text{def}}{=} m \cdot \mathsf{val}_{\mathcal{G}}(\mathbf{x})$ and $y'(\mathbf{x}) \overset{\text{def}}{=} m' \cdot \mathsf{val}_{\mathcal{G}_{\text{spar}}}(\mathbf{x})$

denote the (unnormalized) numbers of constraints satisfied by $\mathbf{x}$ in $\mathcal{G}$ and $\mathcal{G}_{\text{spar}}$, respectively; and $g \overset{\text{def}}{=} m \cdot \mathsf{val}_{\mathcal{G}}(\mathbf{x}) = \max_{\mathbf{x}\in\{0,1\}^n} y(\mathbf{x})$ and $g' \overset{\text{def}}{=} m' \cdot \mathsf{val}_{\mathcal{G}_{\text{spar}}}(\mathbf{x}) = \max_{\mathbf{x}\in\{0,1\}^n} y'(\mathbf{x})$ denote the (unnormalized) values of $\mathcal{G}$ and $\mathcal{G}$', respectively.

For each $e \in E$ let I_e be the indicator random variable for the event that e survives in $\mathcal{G}_{\text{spar}}$. Thus, we can write m' and $y'(\mathbf{x})$ as sums of independent Bern_p random variables:

$$m' = \sum_{(u,v)\in E} I_e \text{ and } y'(\mathbf{x}) = \sum_{(u,v)\in E} I_e x_u(1-x_v).$$

Our first goal is to show that w.h.p., $|g' - pg| \le \epsilon_{\text{spar}} pg/2$.

Lower-bounding g'. Consider any fixed assignment $\mathbf{x}^* \in \{0,1\}^V$ maximizing $\mathcal{G}$'s $\mathsf{Max\text{-}DICUT}$ value, i.e., $g = y(\mathbf{x}^*)$. In particular, $y(\mathbf{x}^*) \ge \frac{1}{4}m$. We have $\mathbb{E}[y'(\mathbf{x}^*)] = py(\mathbf{x}) = pg$ by linearity of expectation, so by the Chernoff bound (Corollary 2.3),

$$\Pr[y'(\mathbf{x}^*) \le (1-\epsilon_{\text{spar}}/2)pg] \le \exp(-\epsilon_{\text{spar}}^2 pg/8).$$

Since $g \ge m/4$, by our assumed bound on p, we get a bound of $\le \exp(-C_{\text{spar}} n/32) = o(1)$. Since $g' \ge y'(\mathbf{x}^*)$, we conclude $g' \ge (1-\epsilon_{\text{spar}}/2)pg$ except with probability $o(1)$.

Upper-bounding g'. To prove an upper bound on g', we take a union bound over all assignments $\mathbf{x} \in \{0,1\}^n$, and prove an $o(2^{-n})$ bound on the probability that $y'(\mathbf{x}) \ge (1+\epsilon_{\text{spar}}/2)pg$.

Fix an assignment $\mathbf{x} \in \{0,1\}^V$. Again, $\mathbb{E}[y'(\mathbf{x})] = py(\mathbf{x})$. We consider two cases depending $y(\mathbf{x})$. If $y(\mathbf{x}) \le m/12$, we have

$$\Pr[y'(\mathbf{x}) \ge (1+\epsilon_{\text{spar}}/2)pg] \le \Pr[y'(\mathbf{x}) \ge pm/4] \le \exp(-pm/8)$$

using $g \ge m/4$ and Corollary 2.4. By our assumed bound on p this is $\exp(-C_{\text{spar}} n/(8\epsilon_{\text{spar}}^2)) = o(2^{-n})$. Otherwise, we have

$$\Pr[y'(\mathbf{x}) \ge (1+\epsilon_{\text{spar}}/2)pg] \le \Pr[y'(\mathbf{x}) \ge (1+\epsilon_{\text{spar}}/2)py(\mathbf{x})] \le \exp(-\epsilon_{\text{spar}}^2 py(\mathbf{x})/12)$$

using $g \ge y(\mathbf{x})$ and Lemma 2.1 (with $\epsilon_{\text{spar}} \le 1$). This is $\le \exp(-C_{\text{spar}} n/144) = o(2^{-n})$ by assumption on $y(\mathbf{x})$ and p. Thus, even taking a union bound over all 2^n assignments, we conclude that $g' \le (1+\epsilon_{\text{spar}}/2)pg$ except with probability $o(1)$.

Bounding m'. Finally, by the Chernoff bound (Corollary 2.3), we know

$$\Pr[|m' - pm| \ge \epsilon_{\text{spar}} pm/2] \le 2\exp(-\epsilon_{\text{spar}}^2 pm/12).$$

By assumption on p, this is at most $2\exp(-C_{\text{spar}} n/12)$ which is $o(1)$.

Putting the bounds together. It remains to prove a bound on $|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{G}_{\text{spar}}}|$. As in Section 6.5, we split into "real" and "normalization" error terms:

$$|\mathsf{val}_{\mathcal{G}} - \mathsf{val}_{\mathcal{G}_{\text{spar}}}| = \left|\frac{1}{m}g - \frac{1}{m'}g'\right| \le \left|\frac{1}{m}g - \frac{1}{pm}g'\right| + \left|\frac{1}{pm} - \frac{1}{m'}\right| g'.$$

The first term can be bounded by $\epsilon_{\text{spar}}/2$ since w.h.p. $|g' - pg| \le \epsilon_{\text{spar}} pg/2$ and $g \le m$. For the second, using $g' \le m'$ the term becomes $|m'/pm - 1| = |m' - pm|/(pm)$, which is at most $\epsilon_{\text{spar}}/2$ since $|m' - pm| \le \epsilon_{\text{spar}} pm/2$. □