Electronic Colloquium on Computational Complexity, Report No. 189 (2023)

Exponential Quantum Space Advantage for Approximating Maximum Directed Cut in the Streaming Model

John Kallaugher
Sandia National Laboratories
jmkall@sandia.gov

Ojas Parekh
Sandia National Laboratories
odparek@sandia.gov

Nadezhda Voronova
Boston University
voronova@bu.edu

Abstract

While the search for quantum advantage typically focuses on speedups in execution time, quantum algorithms also offer the potential for advantage in *space* complexity. Previous work has shown such advantages for data stream problems, in which elements arrive and must be processed sequentially without random access, but these have been restricted to specially-constructed problems [Le Gall, SPAA '06] or polynomial advantage [Kallaugher, FOCS '21]. We show an *exponential* quantum space advantage for the maximum directed cut problem. This is the first known exponential quantum space advantage for any natural streaming problem. This also constitutes the first unconditional exponential quantum resource advantage for approximating a discrete optimization problem in any setting.

Our quantum streaming algorithm 0.4844-approximates the value of the largest directed cut in a graph stream with n vertices using polylog(n) space, while previous work by Chou, Golovnev, and Velusamy [FOCS '20] implies that obtaining an approximation ratio better than $4/9 \approx 0.4444$ requires $\Omega(\sqrt{n})$ space for any classical streaming algorithm. Our result is based on a recent $\widetilde{\mathrm{O}}(\sqrt{n})$ space classical streaming approach by Saxena, Singer, Sudan, and Velusamy [FOCS '23], with an additional improvement in the approximation ratio due to recent work by Singer [APPROX '23].

1 Introduction

Streaming algorithms are a means of processing very large data sets, in particular those too large to be stored wholly in memory. In this setting data elements arrive sequentially, and a streaming algorithm must process elements as they arrive using as little space as possible—ideally logarithmic in the size of the data. Streaming algorithms have been developed for many applications, including computing statistics of data streams and estimating graph parameters [FN85, AMS96, BKS02]. The practical use of such algorithms goes back to the approximate counting algorithm of Morris [Mor78, Fla85] and has expanded as the growth of the internet has increased the prevalence of extremely large datasets [RLS+13].

Quantum computing offers the prospect of exponential resource advantages over conventional classical computers. The primary resource of interest has traditionally been execution time, and while a handful of examples such as Shor's celebrated algorithm for integer factorization [Sho99] provide exponential speedups over the best-known classical counterparts, provable exponential speedups over the best-possible classical algorithms remain largely elusive. Although speedups are important, space is an especially critical resource for quantum computing, as scalable fault-tolerant qubits are and will likely continue to be scarce. Quantum streaming algorithms offer a natural avenue for exploring space-efficient quantum algorithms.

ISSN 1433-8092

Moreover, space-efficient quantum algorithms, including streaming algorithms, offer an alternative opportunity for quantum advantage. Very large datasets continue to be prevalent in computing, and so an algorithm with quantum memory can potentially process much larger datasets than one with only classical memory, provided the problem in question evinces a large enough quantum advantage to justify the much higher cost of qubits relative to classical bits.

Provable exponential space advantages for quantum algorithms in the streaming setting have been known since the seminal work of Gavinsky, Kempe, Kerenidis, Raz, and de Wolf [GKK+08]; however, the problem studied was constructed for the purpose of proving this separation, leaving open the question of whether such advantages exist for problems of independent classical interest. The question of quantum advantage for a "natural" streaming problem was suggested by Jain and Nayak [JN14], who proposed a candidate problem of recognizing the Dyck(2) formal language, related to strings of balanced parentheses. It remains open whether quantum advantage is possible for this problem [NT17].

This question of a quantum advantage for a natural streaming problem was recently resolved in [Kal22], which demonstrates quantum advantage for the problem of counting triangles in graph streams. This problem has been long studied classically and drives numerous applications in analyzing social networks [BKS02, BOV13, JK21]. However, the advantage offered is only *polynomial* in the input size, and requires additional parametrization of the input (the latter being unavoidable with the triangle counting problem). We give the first *exponential* quantum space advantage for a streaming problem of independent interest.

Our Results We give a polylog(n)-space algorithm that gives a 0.4844-approximation for the maximum directed cut (Max-DiCut) problem. Given a directed graph $G = (V, E)$ (as a sequence of edges), Max-DiCut(G) is the greatest number of edges that can be simultaneously "cut" by a partition $V = V_0 \sqcup V_1$, where $\overrightarrow{uv}$ is cut iff $u \in V_0, v \in V_1$. In the classical streaming setting, Chou, Golovnev, and Velusamy [CGV20] showed that any approximation better than 4/9 requires $\Omega(\sqrt{n})$ space.

Theorem 1. *There is a quantum streaming algorithm which* 0.4844*-approximates the* Max-DiCut *value of an input graph G with probability $1-\delta$. The algorithm uses $\mathrm{O}\big(\log^5 n \log \frac{1}{\delta}\big)$ qubits of space.*

Our quantum algorithm is inspired by the classical algorithm of Saxena, Singer, Sudan, and Velusamy [SSSV23a], which achieves this same approximation ratio in a classically-optimal $\widetilde{\mathrm{O}}(\sqrt{n})$ space. Like theirs, our algorithm estimates a histogram of the edges of G where the buckets correspond to edges between "bias classes", which partition the vertices of G according to their bias (the difference between the in-degree and out-degree of a vertex, normalized by the degree). In their algorithm, the histogram is then used to estimate the output value of a 0.4835-approximation algorithm for Max-DiCut due to Feige and Jozeph [FJ15] that randomly assigns each vertex to a side of a cut based solely on its bias. Their algorithm prescribes a constant number of bias ranges, where the random assignment only depends on these ranges. This allows the aforementioned histogram estimate to be constructed with respect to a constant number of bias classes. We use a result from recent work[1] by Singer [Sin23] that improves the approximation ratio offered by [FJ15] with a new set of bias classes, allowing us an improved approximation ratio (we note that this later

[1]The result in [Sin23] also generalizes the result from Max-DiCut to Max-2AND, where the stream consists of arbitrary two-variable clauses and the goal is to approximate the maximum number of them that can be simultaneously satisfied. We believe our algorithm should generalize similarly, but we do not consider the question in this paper.

work also improves the approximation ratio offered by the aforementioned classical algorithm; the advantage of our quantum algorithm lies in its exponentially better space complexity).

The existence of quantum advantage for this problem contrasts with recent work of Kallaugher and Parekh [KP22], who show that approximating the *undirected* maximum cut problem (Max-Cut) in graph streams does not admit any asymptotic quantum advantage. Random assignment of vertices to a side of a cut yields a 1/2-approximation for Max-Cut and a 1/4-approximation for Max-DiCut. These can be implemented as streaming approximations by counting the number of edges and dividing by 2 or 4. While Max-Cut requires linear space (whether in the quantum or classical setting) to do better than a 1/2-approximation, Max-DiCut does admit a $O(\log n)$-space classical algorithm that beats the trivial 1/4-approximation [CGV20]. As noted above, for Max-DiCut, polylog(n)-space classical algorithms cannot offer better than a 4/9-approximation, while we show such a quantum streaming algorithm exists.

Quantum Approximation Advantages Finding provable quantum advantages for approximating discrete optimization problems is an open problem that has received considerable attention, especially following the introduction of the Quantum Approximate Optimization Algorithm (QAOA) [FGG14]. As canonical examples of constraint satisfaction problems (CSPs), Max-Cut and related problems have served as focal points in QAOA analysis and empirical performance studies. The approximability of Max-Cut and other CSPs is well understood conditional on the unique games conjecture (UGC) [Rag08]. If the latter holds, then for every CSP there is some $\alpha \in (0, 1]$ for which an α-approximation is achievable in polynomial time, but for which it is NP-hard to obtain an $(\alpha + \varepsilon)$-approximation for any $\varepsilon > 0$. This leaves little hope for worst-case quantum approximation advantages, as we do not expect polynomial time quantum algorithms to solve NP-hard problems.

Our work, on the other hand, shows that a provable quantum approximation advantage *is* possible in the space-constrained streaming setting. The polylog-space streaming setting does not admit α-approximations of the form described above, which are based on solving semidefinite programs, leaving more room for quantum advantage. For Max-DiCut, $\alpha > 0.874$ is known to be possible in general [LLZ02, BHPZ22], but the work of [CGV20] shows that 4/9 is the best classically possible for polylog-space streaming algorithms. As previously mentioned, it is impossible for a polylog-space quantum algorithm to attain better than a 1/2-approximation for any $\varepsilon > 0$ in the streaming setting. This is also true for Max-DiCut, since approximating Max-Cut can be reduced to approximating Max-DiCut in instances where (v, u) is an edge whenever (u, v) is. Therefore, there is an opportunity for polylog-space quantum β-approximations for $\beta \in (4/9, 1/2]$, and we indeed demonstrate that an exponential quantum space advantage is possible in this range.

2 Our Techniques

We follow the approach of [SSSV23a], who give an $\widetilde{O}(\sqrt{n})$ upper bound for 0.4835-approximation of Max-DiCut. To achieve this result, they use the notion of a "first-order snapshot" of a graph. Introduced in [SSSV23b], this is a histogram of the frequency with which the (directed) edges of the graph go from one "bias class" to another.

The bias of a vertex v is defined as $b_v = \frac{d^{\text{out}} - d^{\text{in}}}{d_v}$ where d_v denotes the degree of v, d_v^{out} denotes the out-degree of v, and d_v^{in} denotes the out-degree of v. Note that $b_v \in [-1, 1]$. Given a partition of $[-1, 1]$ into intervals $[a, b)$ (along with one of the form $[a, 1]$, so that all of $[-1, 1]$ is covered), the *bias*

classes H_i are the sets of of vertices whose biases belong to each interval. The first-order snapshot of the graph is then given by the count of edges $\overrightarrow{uv}$ such that $(u, v) \in H_i \times H_j$ for each (i, j). From hereon we will drop the "first-order" qualification and refer to this simply as a "snapshot".

The key tool used in [SSSV23a] is a result of [FJ15] stating that 0.4835-approximation of the Max-DiCut value of a graph can be computed from its snapshot, where the set of bias classes defining the snapshot is the same for all graphs and so, in particular, the number of different bias classes is constant. We use the result of [Sin23] instead, but the central problem is unchanged: given a constant set of bias thresholds, estimate the corresponding snapshot. See a formal description in Section 4. The authors give a $\widetilde{\mathrm{O}}(\sqrt{n})$-space classical algorithm for estimating these snapshots in the stream, and thereby for computing a 0.4835-approximation of the Max-DiCut value of a graph.

We show that the snapshot of a graph (and thus an approximation to its Max-DiCut value) can be estimated by a quantum streaming algorithm in polylog(n) space. In the remainder of this section, we will describe the main technical ideas behind this contribution.

2.1 One-Way Communication, Simplified Problem

We will start by considering a simplified version of this problem in the one-way communication setting. Suppose there are two parties, Alice and Bob, each with their own input. Alice's input is n labelled graph vertices, and Bob's input is a "directed matching" (a set of vertex-disjoint directed edges), and a pair of labels i, j. Alice is allowed to send a message to Bob, and after receiving this message, Bob's goal is to estimate the number of edges from his matching that have a vertex labeled i as its head and vertex labeled j as its tail. The question in this setting is how small Alice's message could be.

Note that if Alice chooses $\mathrm{O}\big(\sqrt{n}/\varepsilon^2\big)$ vertices uniformly at random and sends the sampled vertices along with the labels, one can show (the "Birthday paradox") that Bob would be able to estimate (up to $\mathrm{O}(\varepsilon n)$ additive error) the number of correct edges with probability 2/3. This is (up to a log factor) optimal for classical protocols by [GKK$^+$08].

On the other hand, with quantum communication, we can achieve a significant improvement using a slightly modified version of the quantum Boolean Hidden Matching protocol from [GKK$^+$08]. Alice sends to Bob k copies of the superposition

$$\frac{1}{\sqrt{2n}}\left(\sum_{v\in V}|vi_vh\rangle + |vi_vt\rangle\right)$$

where i_v is the label of the vertex v, and the last register h, t denotes if the element is to be treated as head or tail of an edge. Bob then measures each copy of the state with the projectors onto the following vectors along with the projector onto the complement of the space they span.

$$\frac{|uih\rangle \pm |vjt\rangle}{\sqrt{2}}$$

for each $\overrightarrow{uv}$ from Bob's graph.

For each copy of the superposition sent and measured, each state of the form $\frac{|uih\rangle+|vjt\rangle}{\sqrt{2}}$ for an edge $\overrightarrow{uv}$ will be returned with probability $1/n$ if u and v are labelled with i and j, with probability $1/4n$ if the vertices have exactly one correct label, and probability zero otherwise. If Bob sees such

an edge, he adds n/k to his estimate of the number of edges with head labelled i and tail labelled j where k is an accuracy parameter to be set later.

Each state of the form $\frac{|uih\rangle - |vjt\rangle}{\sqrt{2}}$ will be returned with probability $1/4n$ if the vertices have exactly one correct label, and probability zero otherwise. If Bob sees such an edge, he subtracts n/k from his estimate.

The expectation of Bob's estimate will therefore be correct, and its variance will be $\mathrm{O}\big(n^2/k\big)$. If $k = \Theta\big(1/\varepsilon^2\big)$, Bob's estimate is εn-close to the correct value with probability 2/3. This protocol only uses $O(\frac{1}{\varepsilon^2}\log n)$ qubits.

2.2 One-way Communication, Snapshot Approximation

Now, let us return to the original problem of estimating the entries of the graph snapshot, while remaining in the two player one-way communication setting. Both Alice and Bob are given a directed graph, and their goal is to estimate the bias histogram of the combined graph.

How is this problem different from the one we considered before? Firstly, Bob's input is no longer a matching, which means the projectors Bob used are no longer guaranteed to be orthogonal. We could address this by splitting Bob's graph into matchings and measuring a different copy of Alice's state with each, but this would require Alice to send $\Theta(d_{\max})$ copies (where $d_{\max}$ is the maximum degree of the graph), which eliminates the advantage we achieved previously[2].

Secondly, we still have to estimate the number of edges between vertices with a pair of labels, but now the labels (bias classes) depend on the graph itself. Moreover, neither Alice nor Bob alone know the labels since the labels depend on both Alice and Bob's input graphs.

Fortunately, we can tackle both problems using properties of how the biases depend on the two players' graphs. Our first observation is that, if we're given the biases of a vertex in Alice's input and Bob's input separately, as well as the degrees of this vertex in each of the graphs, we can obtain its bias with respect to the whole graph. The second observation is that if the degree of a vertex in Bob's graph is much higher than its degree in Alice's graph, Alice's graph contributes very little to the bias of this vertex and thus Bob can compute an estimate of the bias by himself.

The former means that, if Alice sketches her vertices in polylog(n) different subsets based on their degrees and biases (so that for each sketch, Bob knows the degree and bias of the vertices he is dealing with up to a small error), Bob has enough information to approximate the biases in the full graph. The latter fact means that, if Alice copies each of her vertices with multiplicity equal to a sufficiently large constant times its degree (which she can afford to do, as it results in a superposition with no more than $\mathrm{O}(m)$ states, where m is the number of edges in her input), either Bob will be able to measure with all of his edges, or Bob does not need Alice's input to determine the bias of this vertex.

Suppose Alice sends the superposition

$$\frac{1}{\sqrt{|H|+|T|}} \sum_{v\in H} \sum_{i\in[\lceil d_H/\varepsilon\rceil]} |vih\rangle + \sum_{v\in T} \sum_{i\in[\lceil d_T/\varepsilon\rceil]} |vit\rangle$$

where H, T are subsets of Alice's vertices with biases and degrees in a small enough range (such that Bob can know the biases and degrees to an ε approximation), with d_H, d_T upper bounds on

[2]This same issue arises in [Kal22]. There it was solved by using a classical algorithm to handle cases where $d_{\max}$ is particularly bad, at the cost of reducing the exponential advantage to only polynomial.

the degrees of vertices in H and T. Bob may then measure the state with the projectors onto the vectors

$$|ui_{\overrightarrow{uv}}t\rangle \pm |vj_{\overrightarrow{uv}}t\rangle$$

for each $\overrightarrow{uv}$ in Bob's graph that has the degree of u in Bob's graph at most d_H/ε and the degree of v in Bob's graph at most d_T/ε. The notation $i_{\overrightarrow{uv}}$ denotes the index of edge $\overrightarrow{uv}$ in a fixed ordering of out-edges of u, and $j_{\overrightarrow{uv}}$ denotes the index of edge $\overrightarrow{uv}$ in a fixed ordering of in-edges of v. Now the measurement operators are orthogonal again.

Similarly to the previous case, $|ui_{\overrightarrow{uv}}t\rangle + |vj_{\overrightarrow{uv}}t\rangle$ will be the measurement outcome with probability $2/(|H| + |T|)$ if $u \in H$ and $v \in T$, and $|ui_{\overrightarrow{uv}}t\rangle \pm |vi_{\overrightarrow{uv}}t\rangle$ are equally likely otherwise. So Bob can use this measurement to estimate how many of his edges go from H to T, and also to *sample* from such edges[3].

Given such a sampled edge, Bob can calculate the bias of its endpoints using his knowledge of the degrees and biases of u and v restricted to Alice's graph, and their degrees and biases in his own graph. So by repeating this process for polylog(n) many combinations of ranges, the players can approximate the number of edges between each pair of "bias classes", with the exception of vertices whose degree is much higher (more than $1/\varepsilon$ times as high) for Bob than Alice.

For these vertices, Bob can approximate their contribution to the snapshot with only his own information, as Alice's input only provides an ε contribution to their biases. Of course, he does not actually know which vertices these should be. However, this can be solved by having Bob calculate this for *every* vertex, and then performing appropriate corrections when the protocol samples non low-degree vertices.

2.3 Streaming Algorithm

Now we want to implement this protocol in the stream. Our first obstacle is that now, each time a new edge arrives, we need to "process" it both as Alice and as Bob, meaning we need to update the superposition and measure it. Secondly, instead of performing our measurements all at once, we will need to perform them edge by edge, at each point using projectors onto a small part of the space along with a "complementary projector" onto the rest of the space. When a measurement returns something *other* than the complementary projector, our quantum stage will terminate and we will proceed classically, but when the complementary projector is returned it will be important to ensure that the state retains the properties needed for future measurements. Finally, we may no longer simply restrict our input to vertices in a specific class of degrees or biases, as at the time we see an edge we do not know what edges have arrived incident to its endpoints in the past (let alone the future). So we need to maintain a quantum state that somehow encodes the degree and bias of each vertex, in a way such that whenever an edge arrives, we can measure with it and have zero (expected) effect on our output when the edge's endpoints have the wrong degrees or biases.

Start by considering the case where we are only interested in whether the endpoints of the edge have the right *degree*. To make things even simpler, imagine we only care whether a vertex has degree *at least* d. We may store the state

$$\frac{1}{\sqrt{M'}}\left(\mathcal{S} + \sum_{v\in V}\sum_{i=1}^{[d_v]}(|vih\rangle + |vit\rangle)\right)$$

[3]With the caveat that he will also sample pairs that don't exist. But because each fictitious pair is seen equally often with a + or − sign, he can still sample "in expectation"

where d_v is the degree of v among the edges that have arrived up until now, and $\mathcal{S}$ is is a collection of "scratch states" that do not contain any information about the graph and are there to be swapped with states we want, in order to maintain this superposition. When a new edge $\overrightarrow{uv}$ arrives, we can maintain this state by sending $|uih\rangle \to |u(i+1)h\rangle$, $|uit\rangle \to |u(i+1)t\rangle$,$|vih\rangle \to |v(i+1)h\rangle$,$|vit\rangle \to |v(i+1)t\rangle$ and swapping[4] states from $\mathcal{S}$ for $|u1h\rangle, |u1t\rangle, |v1h\rangle, |v1t\rangle$. $\mathcal{S}$ will therefore require $2m$ states to start (where m is the number of edges in the stream) and so the normalization factor M' will start at $M = 2m$, before changing as the state is measured.

To use this state, after $\overrightarrow{uv}$ arrives and the state has been updated accordingly, we may measure with the projectors (along with a complementary projector onto the space not spanned by them) onto the vectors

$$|ud_Hh\rangle \pm |vd_Tt\rangle$$

where d_H, d_T are the minimum desired degrees for the head and the tail respectively. Then, if both endpoints have achieved the desired degree, $|ud_Hh\rangle + |vd_Tt\rangle$ is a possible outcome for the measurement but not $|ud_Hh\rangle - |vd_Tt\rangle$. Otherwise both are equally likely, and so we can estimate the number of edges between pairs of vertices with the right degrees (similar to the 2-player labeling problem). Note that if $|ud_Hh\rangle \pm |vd_Tt\rangle$ *is* returned, the state collapses to the corresponding vector—at this point we stop using the state and proceed classically with the returned vertices u and v (e.g. counting how many in- and out-edges are seen incident to them after this point). Therefore, each copy of the quantum state maintained allows us to "sample" up to one pair of vertices (u, v).

One side-effect of this measurement, that will be convenient later, is that conditioned on returning the complementary projector (i.e. not sampling some (u, v) and terminating the quantum stage) it "deletes" $|ud_Hh\rangle$ and $|vd_Tt\rangle$ from the superposition after each measurement, so the state we maintain ends up being

$$\frac{1}{\sqrt{M'}}\left(\mathcal{S} + \sum_{v\in V}\sum_{i=1}^{\min(d_v, d_H-1)} |vih\rangle + \sum_{i=1}^{\min(d_v, d_T-1)} |vit\rangle\right).$$

This also means that each measurement before termination reduces M', and so the probability of returning a given measurement outcome does not depend on how many edges have been processed so far (as the probability that the algorithm terminates before a given edge is processed exactly cancels out the decrease in M' conditioned on the algorithm *not* terminating before then). We can extend this method to only count (in expectation) edges where u and v are in *ranges* $[d_H, d'_H)$ and $[d_T, d'_T)$, by maintaining four copies of the state and measuring all the combinations of d_H, d'_H and d_T, d'_T, and subtracting appropriately. This means we are now maintaining four *different* states for each sample we want, because of the difference in which states the measurements delete:

$$\frac{1}{\sqrt{M'}}\left(\mathcal{S} + \sum_{v\in V}\sum_{i=1}^{\min(d_v, x-1)} |vih\rangle + \sum_{i=1}^{\min(d_v, y-1)} |vit\rangle\right)$$

where $x = d_H$ or d'_H and $y = d_T$ or d'_T.

Unfortunately, the information only about the degrees is not enough to estimate the bias. Moreover, our method does not automatically permit associating tuples of integers with each vertex. Our approach is to store the out-degree of the vertex in the higher order bits of the degree counter.

[4]That is, executing the unitary operation that swaps a given basis element $|is\rangle$ in the superposition $\mathcal{S} = \sum_i |is\rangle$ for the desired state while leaving the rest of the space unchanged. Note that this is distinct from the "swap" operation that e.g. swaps the two registers of a two-qubit state. By maintaining a single counter we may track how many of these "scratch states" have already been used and thus ensure our swap always targets a state present in the superposition.

To do this without overwriting the degree information[5], we will encode seeing an edge $\overrightarrow{uv}$ by sending $|uih\rangle \to |u(i+d'_H)h\rangle$ and $|uit\rangle \to |u(i+d'_T)t\rangle$, and then swapping out $d'_H + d'_T$ scratch states for $|u1h\rangle \dots |ud'_H h\rangle$ and $|u1t\rangle \dots |ud'_T t\rangle$. In order to avoid blowing up the number of states we need, we only do this with probability $1/d_H$, $1/d_T$ respectively (we will always choose d_H and d'_H to be within a constant factor of each other), and so if, for instance, we see $d_u - 1$ edges incident to a vertex u that do not trigger this event, and then see one that does trigger it, the $|uih\rangle$ portion of our state becomes

$$\sum_{i=1}^{\min(d_u, d_H-1)} |uih\rangle + \sum_{u=d_H}^{d_u+d'_H} |uih\rangle$$

up to normalization. Now, if we measure with

$$|u(d_H + d'_H)h\rangle \pm |v(d_T + d'_T)t\rangle$$

in addition to

$$|u(d_H)h\rangle \pm |v(d_T)t\rangle$$

we can get an estimator that counts when both u and v have degree at least d_H and d_T respectively, *and* have each seen at least once out-edge that was sampled. So this approximately checks the out-degree of u and v, and as before we can convert this into approximately counting how often these out-degrees lie in certain *ranges* by adding three additional estimators.

This method only checks a very rough approximation to the out-degrees and therefore biases of u and v. We improve it by adding extra estimators

$$|u(d_H + id'_H)h\rangle \pm |v(d_T + jd'_T)t\rangle$$

for $i = 1, \dots, \kappa - 1$ and $j = 1, \dots, \kappa - 1$. This has κ^2 overhead as we need the measurements to be orthogonal, and so need to perform them on different pairs of states, but we only need constant κ for a sufficiently accurate estimate. In combination with some "cleanup" measurement operators we end up with e.g. the $|uih\rangle$ portion of the state being

$$\sum_I \sum_{i \in I} |uih\rangle$$

up to normalization, where the I are a sequence of intervals, with the last element in each interval encoding the degree of u and the *number* of intervals encoding the out-degree of the graph.

Dealing with noise in the bias The final issue is that our estimate of the bias of a vertex is noisy, because of the way we count out-edges. One challenge for snapshot algorithms is that even a small error in estimating the bias of a vertex can lead to large errors in the snapshot, if e.g. a vertex with large degree has bias close to the boundary of a class. The authors of [SSSV23a] address this problem by "smoothing" techniques. We, on the other hand, introduce an object that we call the "pseudosnapshot", corresponding to our noisy estimates of the biases, and show that approximating the entries of this object suffices for the Max-DiCut approximation.

[5] One might ask why we need to use d'_H and d'_H here, as d_H and d_T would suffice to avoid a conflict with the degree information. This will become necessary because of how our measurements delete states from the superposition.

2.4 Proof overview

In section 4 we recall the statement of the reduction from Max-DiCut to computing snapshots. In section 5 we introduce the notion of the pseudosnapshot and how it is related to the snapshot. In section 6 we describe the key quantum primitive that estimates each entry of the pseudosnapshot (restricted to edges with head and tail in specific ranges of degrees) and prove its correctness. Finally, in section 7 we show how to put everything together to get the algorithm for 0.4844-approximation of Max-DiCut.

3 Preliminaries

The graphs we deal with will all be directed graphs. When the graph $G = (V, E)$ being dealt with is clear, n will be the number of the vertices of the graph, and m the number of edges. For all $v \in V$, d_v is its degree, d_v^{out} the number of edges with v as their head, d_v^{in} the number with v as their tail, and $b_v = \frac{d_v^{\text{out}} - d_v^{\text{in}}}{d_v}$ will be the *bias* of v.

We will be interested in the *maximum directed cut value* of a graph.

Definition 2. Let $G = (V, E)$ be a directed graph. Then

$$\text{Max-DiCut}(G) = \max_{x \in \{0,1\}^V} |\{\overrightarrow{uv} \in E : x_u = 0, x_v = 1\}|.$$

Specifically, we will be interested in the complexity of attaining an α-approximation to Max-DiCut.

Definition 3. For any $X, X' \in \mathbb{R}_{\geq 0}$, X' α-approximates X if $X' \in [\alpha X, X]$.

Note that recent work on streaming Max-Cut [KK19, KP22] uses the opposite definition, where $X' \in [X, \alpha X]$, so the complexity of achieving an α-approximation for us is equivalent to their achieving a $1/\alpha$ approximation. We adopt this notation for consistency with recent work on streaming Max-DiCut [SSSV23a].

4 Reducing Max-DiCut to Snapshot Estimation

We follow [SSSV23a] in reducing this problem to the problem of estimating a "snapshot" of the graph.

Definition 4. Let $\mathbf{t} \in [-1, 1]^\ell$ be a vector of bias thresholds. The (first-order) snapshot $\mathsf{Snap}^G \in \mathbb{N}^{\ell \times \ell}$ of $G = (V, E)$ is given by:

$$\mathsf{Snap}^G_{i,j} = |\{\overrightarrow{uv} \in E\} : u \in H_i, v \in H_j|$$

where H_i is the i^{th} "bias class", given by

$$H_i = \begin{cases} \{v \in V : b_v \in [\mathbf{t}_i, \mathbf{t}_{i+1})\} & i \in [\ell - 1] \\ \{v \in V : b_v \in [\mathbf{t}_\ell, 1]\} & i = \ell. \end{cases}$$

Lemma 5 (From Table 1 of [Sin23], strengthening of Lemma 3.19 of [SSSV23a], in turn from [FJ15])**.** *There exists a constant $\alpha > 0.4844$, $\ell \in \mathbb{N}$, a vector of bias thresholds $\mathbf{t} \in [-1, 1]^\ell$, and a vector of probabilities $\mathbf{r} \in [0, 1]^\ell$ such that $\mathbf{r}^\dagger \mathsf{Snap}^G (1^\ell - \mathbf{r})$ is an α-approximation to the* Max-DiCut *value of G.*

5 Reducing Max-DiCut to Pseudosnapshot Estimation

In this section, we define a "pseudosnapshot" PsSnap^G, based on hash functions and a coarsening of the vertex degrees, and show that it is close to the snapshot of a graph G' that is in turn close to G. In the next section, we will show that this can be approximated by a small space quantum streaming algorithm.

5.1 Definition

Let $(d_i)_{i=0}^{\lfloor \log_{1+\varepsilon^3} n \rfloor}$ be given by $d_i = \lfloor (1+\varepsilon^3)^i \rfloor$ for $i < \lfloor \log_{1+\varepsilon^3} n \rfloor$ and $d_{\lfloor \log_{1+\varepsilon^3} n \rfloor} = n$. Let $\kappa \leq \operatorname{poly} n$, $\varepsilon \in [0,1]$ be accuracy parameters to be chosen later, and let $(f_i)_{i=0}^{\lfloor \log_{1+\varepsilon^3} n \rfloor}$ be a family of fully independent random hash functions such that $f_i : E \to \{0,1\}$ is 1 with probability $\kappa/2d_i$, while $g : V \to [-\varepsilon, \varepsilon]$ is a fully independent random hash function that is uniform on $[-\varepsilon, \varepsilon]$.[6]

Fix an arrival order for the edges e of the directed graph. For any vertex v and edge e, let $d_v^{\text{out},\leq e}$, $d_v^{\leq e}$, refer to the out-degree and degree of v when only e and edges that arrive before e are counted, and let $d_v^{\text{out},>e}$, $d_v^{>e}$ refer to these quantities when counting only edges that arrive *after* e. Let $\widetilde{i}$ be the largest i such that $d_i < d^{\leq e}$. Then define $\widetilde{d}_v^{\leq e} = d_{\widetilde{i}}$, and let $\widetilde{d}_v^{\text{out},\leq e}$ be the number of edges e' with head v that arrive before e and have $f_{\widetilde{i}}(e') = 1$, multiplied by $2d_{\widetilde{i}}/\kappa$. We will then define the e-pseudobias of v, $\widetilde{b}_v^e$, as

$$\min\left\{2\frac{\widetilde{d}_v^{\text{out},\leq e} + d_v^{\text{out},>e}}{\widetilde{d}_v^{\leq e} + d_v^{>e}} - 1 + g(v), 1\right\}.$$

In other words, the $\widetilde{b}_v^e$ is the bias of v when its degree among e and edges that arrive before e is rounded to the bottom of the interval $[d_{\widetilde{i}}, d_{\widetilde{i}+1})$, and its out-degree among these edges is estimated using the number of out-edges "sampled" by $f_{\widetilde{i}}$, with a small amount of noise $g(v)$ added. Since this can sometimes produce a pseudobias larger than 1, we then cap it at 1.

Definition 6. Let $\mathbf{t} \in [-1,1]^\ell$ be a vector of bias thresholds. The pseudosnapshot $\mathsf{PsSnap}^G \in \mathbb{N}^{\ell \times \ell}$ of $G = (V, E)$ is given by:

$$\mathsf{PsSnap}_{i,j}^G = |\{\overrightarrow{uv} \in E : u \in H_i^{\overrightarrow{uv}}, v \in H_j^{\overrightarrow{uv}}\}|$$

where H_i^e is the i^{th} "e-pseudobias" class, given by

$$H_i^e = \begin{cases} \{v \in V : \widetilde{b}_v^e \in [\mathbf{t}_i, \mathbf{t}_{i+1})\} & i \in [\ell - 1] \\ \{v \in V : \widetilde{b}_v^e \in [\mathbf{t}_\ell, 1]\} & i = \ell. \end{cases}$$

The restriction of PsSnap^G to $E' \subseteq E$ is then given by:

$$\mathsf{PsSnap}_{i,j}^{G,E'} = |\{\overrightarrow{uv} \in E' : u \in H_i^{\overrightarrow{uv}}, v \in H_j^{\overrightarrow{uv}}\}|$$

[6] We adopt this notation for the sake of clarity, but we will only ever evaluate $f_i(e)$ at the update when edge e arrives, and g after processing the entire stream on the endpoints of edges our algorithm stores, so we do not need to pay the prohibitive overhead of storing these hash functions. Moreover, while we write $g(e)$ as a random real number, it will only ever be used in sums and comparisons with numbers of $\operatorname{poly}(n, \varepsilon)$ precision, so we do not need to store it any more precision than that.

5.2 Closeness to Snapshot

In this section we show that the snapshot and pseudosnapshot of a graph are close enough to each other for the purpose of approximating Max-DiCut. We will assume throughout that the snapshot and pseudosnapshot are defined relative to the same vector of bias thresholds $\mathbf{t}$. We will refer to the intervals $([\mathbf{t}_i, \mathbf{t}_i))_{i\in[\ell-1]}$ and $[\mathbf{t}_\ell, 1)$ as "bias intervals".

We start by showing that, with good enough probability, most of the edges incident to any vertex v will give v a pseudobias in the same interval, and that pseudobias will not be too far from the true bias.

Lemma 7. *For each $v \in V$, with probability $1 - \mathrm{O}(\varepsilon^2) - e^{-\mathrm{O}(\varepsilon^6\kappa)}$ over the hash functions $(f_i)_{i=0}^{\lfloor \log_{1+\varepsilon^3} n\rfloor}$ and g, $\widetilde{b}_v^e$ is in the same bias interval for all but εd_v of the edges e incident to v, and all of these $\widetilde{b}_v^e$ are within $\mathrm{O}(\varepsilon)$ of b_v. Moreover, these events depend only on $g(v)$ and $f(e)$ for edges with head v.*

Proof. As there are only constantly many bias intervals, we may without loss of generality assume that ε is smaller than half the distance between the boundaries of any bias interval. Then for every $v \in V$, with probability $1-\mathrm{O}(\varepsilon^2)$ over $g(v)$, $b_v+g(e)$ is at least $C\varepsilon^3$ away from the boundary of any bias interval, for $C > 0$ a constant to be chosen later. It will therefore suffice for $|\widetilde{b}_v^e - b_v - g(e)| < C\varepsilon^3$ to hold for all but εd_v of the edges e incident to v.

First note that, by the definition of the d_i, $\widetilde{d}_v^{\le e} \in \left[\frac{d_v^{\le e}}{1+\varepsilon^3}, d_v^{\le e}\right)$ for all e. So we only need to bound $\widetilde{d}_v^{\mathrm{out},\le e}$.

We will ignore the first εd_v edges to arrive incident to v. For the edges e arriving after this, $d_v^{\le e}$ passes through only $\mathrm{O}\left(\log_{1+\varepsilon^3}\frac{1}{\varepsilon}\right) = \mathrm{O}\left(\frac{1}{\varepsilon^3}\log\frac{1}{\varepsilon}\right)$ intervals $[d_i, d_{i+1})$. It will suffice to show that, for each such interval,

$$\widetilde{d}_v^{\mathrm{out},\le e} \in ((1-\varepsilon^3)d_v^{\mathrm{out},\le e}, (1+\varepsilon^3)d_v^{\mathrm{out},\le e})$$

for the d_i^{th} e to arrive incident to v (as $\widetilde{d}_v^{\mathrm{out},\le e}$ only increases, and $d_v^{\mathrm{out},\le e}$ only changes by a $(1+\varepsilon^3)$ multiplicative factor in this interval).

So fix such an e. Then $\widetilde{d}_v^{\mathrm{out},\le e}$ is $2d_i/\kappa$ times the number of edges e' that arrive before e (including e) and have $f_i(e') = 1$. So it is distributed as

$$\frac{2d_i}{\kappa}\mathrm{B}\left(d_v^{\mathrm{out},\le e}, \frac{\kappa}{2d_i}\right)$$

and so by the Chernoff bounds it is within $\varepsilon^3 d_v^{\mathrm{out},\le e}$ of $d_v^{\mathrm{out},\le e}$ with probability

$$1 - e^{-\mathrm{O}\left(\varepsilon^6\kappa\frac{d_v^{\mathrm{out},\le e}}{d_i}\right)} = -e^{-\mathrm{O}(\varepsilon^6\kappa)}$$

and by taking a union bound over the $\mathrm{O}\left(\frac{1}{\varepsilon^3}\log\frac{1}{\varepsilon}\right)$ intervals, we have that, with probability $1 - e^{-\mathrm{O}(\varepsilon^6\kappa)}$ over $(f_i)_{i=0}^{\lfloor \log_{1+\varepsilon^3} n\rfloor}$, $\widetilde{d}_v^{\le e}$ and $\widetilde{d}_v^{\mathrm{out},\le e}$ are $(1+\mathrm{O}(\varepsilon^3))$ multiplicative approximations of $d_v^{\le e}$ and $d_v^{\mathrm{out},\le e}$, respectively, and so

$$\widetilde{b}_v^e \in (b_v + g(v) - \mathrm{O}(\varepsilon^3), b_v + g(v) + \mathrm{O}(\varepsilon^3))$$

for all e incident to v after the first εd_v to arrive. The lemma therefore holds if we choose C to be a large enough constant. □

This allows us to show that, with only small edits to G, we can change the bias of vertices in G in such a way that the pseudosnapshot becomes an approximately accurate snapshot.

Lemma 8. *With probability* $1 - \mathrm{O}(\varepsilon) - e^{-\mathrm{O}(\varepsilon^6\kappa)}$ *over* $(f_i)_{i=0}^{\lfloor \log_{1+\varepsilon^3} n \rfloor}$ *and* g*, there exists a graph* G' *differing from* G *in* $\mathrm{O}(\varepsilon m)$ *edges such that any* $\mathrm{O}(\varepsilon m)$*-accurate estimate of the pseudosnapshot of* G *is a* $\mathrm{O}(\varepsilon m)$*-accurate estimate of the snapshot of* G'*.*

Proof. Let V' be the set of all vertices V for which the event described in Lemma 7 occurs. The vertex set of G' will be $V' \cup \{\perp\}$, where $\perp$ is a newly introduced dummy vertex. We will construct G' by first removing all vertices in $V \setminus V'$ from G, and replacing the edges between them and V' with corresponding edges between $\perp$ and V'.

Then, for each vertex $v \in V'$, we will add $\mathrm{O}(\varepsilon d_v)$ edges between v and $\perp$, choosing the orientation of these edges so that the bias of v in G' is in the same bias interval as $\widetilde{b}_v^e$ for all but εd_v of the edges e incident to v.

G' differs from G in at most

$$2 \sum_{v \in V \setminus V'} d_v + \sum_{v \in V'} \mathrm{O}(\varepsilon d_v)$$

edges. So by Markov's inequality, as each edge is in $V \setminus V'$ with probability at most $\mathrm{O}(\varepsilon^2) + e^{-\mathrm{O}(\varepsilon^6\kappa)}$, this is $\mathrm{O}(\varepsilon m)$ with probability at least $1 - \mathrm{O}(\varepsilon) - e^{-\mathrm{O}(\varepsilon^6\kappa)}$.

Moreover, by Lemma 7 all but $\sum_{v \in V \setminus V'} d_v + \sum_{v \in V} \varepsilon d_v$ of the edges counted in the pseudosnapshot of G had endpoints whose pseudobiases were in the same bias intervals as the biases of the corresponding vertices in G'. Furthermore, there are only $\sum_{v \in V \setminus V'} d_v + \sum_{v \in V'} \mathrm{O}(\varepsilon d_v)$ edges in G' that were *not* counted in the pseudosnapshot of G—those added to replace edges between V and V', and those added to correct the biases of edges in V'. So again by Markov, the pseudosnapshot of G is a $\mathrm{O}(\varepsilon m)$-accurate snapshot for G' with probability at least $1 - \mathrm{O}(\varepsilon) - e^{-\mathrm{O}(\varepsilon^6\kappa)}$. □

5.3 Reduction from Max-DiCut

Now, we can show that the pseudosnapshot gives a good Max-DiCut estimate, by using the fact that the G' constructed in the previous section has a similar Max-DiCut value to G.

Lemma 9. *Let* α*,* ℓ*,* $\mathbf{t}$*,* $\mathbf{r}$ *be as in Lemma 5. Then there exists a constant* $C > 0$ *such that*

$$\mathbf{r}^\dagger \mathsf{PsSnap}^G(1^\ell - \mathbf{r}) - C\varepsilon m$$

is an $(\alpha - \mathrm{O}(\varepsilon))$*-approximation to* Max-DiCut(G) *with probability* $1 - \mathrm{O}(\varepsilon) - e^{-\mathrm{O}(\varepsilon^6\kappa)}$ *over* $(f_i)_{i=0}^{\lfloor \log_{1+\varepsilon^3} n \rfloor}$ *and* g*.*

Proof. Suppose the probability $1 - \mathrm{O}(\varepsilon) - e^{-\mathrm{O}(\varepsilon^6\kappa)}$ event of Lemma 8 holds. Then there exists a graph G' that differs from G in only εm edges such that

$$|\mathsf{PsSnap}^G - \mathsf{Snap}^{G'}| = \mathrm{O}(\varepsilon m)$$

and so

$$|\mathbf{r}^\dagger \mathsf{PsSnap}^G(1^\ell - \mathbf{r}) - \mathbf{r}^\dagger \mathsf{Snap}^{G'}(1^\ell - \mathbf{r})| = \mathrm{O}(\varepsilon m)$$

which by Lemma 5 implies that

$$\alpha\textsc{Max-DiCut}(G') - \mathrm{O}(\varepsilon m) \le \mathbf{r}^\dagger \mathsf{PsSnap}^G(1^\ell - \mathbf{r}) \le \textsc{Max-DiCut}(G') + \mathrm{O}(\varepsilon m).$$

As G' only differs from G in $\mathrm{O}(\varepsilon m)$ edges,

$$|\textsc{Max-DiCut}(G') - \textsc{Max-DiCut}(G)| = \mathrm{O}(\varepsilon m)$$

and so the result follows by setting C to be a large enough constant that

$$\mathbf{r}^\dagger \mathsf{PsSnap}^G(1^\ell - \mathbf{r}) - C\varepsilon m \le \textsc{Max-DiCut}(G).$$

□

6 Quantum Algorithm for Pseudosnapshot Estimation

In this section we describe our key quantum primitive: a small-space algorithm for estimating the pseudosnapshot of a directed graph $G = (V, E)$, restricted to edges $e = \overrightarrow{uv}$ with $d_u^{\le e} \in [d_i, d_{i+1})$, $d_v^{\le e} \in [d_j, d_{j+1})$ for some specified $i, j \in [\lfloor \log_{1+\varepsilon^3} n \rfloor] \cup \{0\}$.

6.1 Algorithm

Lemma 10. *Let $\alpha, \beta \in [\lfloor \log_{1+\varepsilon^3} n \rfloor] \cup \{0\}$. Let $\kappa = \mathrm{poly}(n)$ be the integer accuracy parameter used in defining the pseudobiases. Fix a draw of the hash functions $(f_i)_{i=0}^{\lceil \log_{1+\varepsilon^3} n \rceil}$, and therefore the pseudobiases of the graph G.*

Then there is a quantum streaming algorithm that, if $\sum_{e \in E}(f_\alpha(e) + f_\beta(e)) \le 2\kappa m$, returns an estimate of the pseudosnapshot of G restricted to edges $e = \overrightarrow{uv}$ with $d_u^{\le e} \in [d_\alpha, d_{\alpha+1})$, $d_v^{\le e} \in [d_\beta, d_{\beta+1})$.

Each entry of the estimate has bias at most the number of edges $\overrightarrow{uv}$ such that:

1. $d_u^{\le \overrightarrow{uv}} \in [d_\alpha, d_{\alpha+1})$
2. $d_v^{\le \overrightarrow{uv}} \in [d_\beta, d_{\beta+1})$
3. $\max\left\{ \frac{\kappa}{2d_\alpha} \widetilde{d}_u^{out, \le \overrightarrow{uv}} + 1, \frac{\kappa}{2d_\beta} \widetilde{d}_v^{out, \le \overrightarrow{uv}} + 1 \right\} > \kappa$

Each entry has variance $\mathrm{O}\big(\kappa^3 m^2\big)$. The algorithm uses $\mathrm{O}(\log n)$ qubits of space.

The algorithm will maintain a superposition

$$\frac{|\mathcal{S}\rangle + \sum_{i=1}^{2\kappa^2} \big(|\mathcal{A}^i\rangle + |\mathcal{B}^i\rangle + |\mathcal{C}^i\rangle + |\mathcal{D}^i\rangle \big)}{\sqrt{M'}}$$

where $M' \le M = C\kappa^3 m$ for some sufficiently large constant C. M' will start at M and decrease as measurements remove states from the superposition.

A key primitive for maintaining this state will be "swapping", in which we execute the unitary that swaps two named basis states while leaving the rest of the Hilbert space unchanged.

$|\mathcal{S}\rangle$ will be our "scratch states". It is equal to

$$\sum_{i=t}^{M} |is\rangle$$

where t starts at 1 and is incremented every time we use a scratch state (by swapping it with some other state we want), and the last register s indicates that this is a scratch state. The algorithm will keep track of t so that it knows which state to swap from.

The remaining vectors encode information about vertices and their degrees. For $\mathcal{E} = \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$, each takes the form

$$|\mathcal{E}^i\rangle = \sum_{v \in V} |\mathcal{E}^i_v\rangle$$
$$|\mathcal{E}^i_v\rangle = \sum_{j \in E_v} |vje\rangle$$

where the sets $E_v = A_v, B_v, C_v, D_v$ are not explicitly stored by the algorithm but are updated by updating the superposition, and $e = a^i, b^i, c^i, d^i$ marks which of $\mathcal{A}^i, \mathcal{B}^i, \mathcal{C}^i, \mathcal{D}^i$ a state belongs to, for $i \in [2\kappa^2]$. Note that the underlying set E_v for each $\mathcal{E}^i_v$ does not depend on i—the $2\kappa^2$ copies are to allow us to perform a larger set of measurements, and the states will remain identical until the algorithm terminates. The four non-scratch components are broken into two pairs (as each vertex can be both a head and a tail, and the two cases need to be handled separately), with each pair having an element for checking whether degrees are high enough, and one for checking whether degrees are low enough (both will track whether vertices have had enough edges coming out of them that pass the hash functions).

- $\mathcal{A}^i$ are for tracking vertices with degree at least d_α.
- $\mathcal{B}^i$ are for tracking when those vertices have their degree exceed $d_{\alpha+1}$.
- $\mathcal{C}^i$ are for tracking vertices with degree at least d_β.
- $\mathcal{D}^i$ are for tracking when those vertices have their degree exceed $d_{\beta+1}$.

At the start of the execution of our algorithm, $A_v, B_v, C_v, D_v = \emptyset$. They will be updated with the following three operations:

- $\texttt{inc}(\mathcal{E}, v, r)$ replaces E_v with $\{i + r : i \in E_v\} \cup [r]$, and replaces t with $t + 2\kappa^2 r$. Note that this can be accomplished with a single unitary transformation, by sending $|vie^j\rangle$ to $|v(i + r)e^j\rangle$ for each i, j, and then swapping the first $2\kappa^2 r$ remaining elements of $\mathcal{S}$ with $\sum_{i=1}^{r} |vie^j\rangle$ for each $j \in [2\kappa^2]$.
- $\texttt{measure}(u, v)$ measures the superposition with projectors onto the following vectors for each $(i, j) \in [\kappa]^2$, and $b \in \{0, 1\}$, along with a projection onto the complement of the space spanned

by the projectors.

$$|\nu^1_{i,j,b}\rangle = \frac{|u(d_\alpha + (i-1)d_{\alpha+1})a^{r_{i,j}}\rangle + (-1)^b|v(d_\beta + (j-1)d_{\beta+1})c^{r_{i,j}}\rangle}{\sqrt{2}}$$
$$|\nu^2_{i,j,b}\rangle = \frac{|u(id_{\alpha+1})b^{r_{i,j}}\rangle + (-1)^b|v(d_\beta + (j-1)d_{\beta+1})c^{s_{i,j}}\rangle}{\sqrt{2}}$$
$$|\nu^3_{i,j,b}\rangle = \frac{|u(d_\alpha + (i-1)d_{\alpha+1})a^{s_{i,j}}\rangle + (-1)^b|v(jd_{\beta+1})d^{r_{i,j}}\rangle}{\sqrt{2}}$$
$$|\nu^4_{i,j,b}\rangle = \frac{|u(id_{\alpha+1})b^{s_{i,j}}\rangle + (-1)^b|v(jd_{\beta+1})d^{s_{i,j}}\rangle}{\sqrt{2}}$$

Where the $s_{i,j}$, $r_{i,j}$ are chosen so that these vectors are all orthogonal to one another across all i, j, b (note that this is possible because we have κ^2 possible values to choose from). If the result of this measurement is anything other than the complementary projection, the quantum part of the algorithm will terminate and the remaining execution will be entirely classical.

- `cleanup`(u, v) measures the superposition with projectors onto $|w(d_\alpha + id_{\alpha+1})a^j\rangle$, $|w((i+1)d_{\alpha+1})b^j\rangle$, $|w(d_\beta + id_{\beta+1})c^j\rangle$, and $|w((i+1)d_{\beta+1})d^j\rangle$, for all $i \in [M]$, $j \in [2\kappa^2]$, and $w = u, v$, along with a projector onto the complement of the space they span. If anything other than the complementary projector is returned, the algorithm halts entirely and outputs a zero estimate for the pseudosnapshot.

  Together, the effect of performing `measure`(u, v) and `cleanup`(u, v), if they do *not* return something other than the complementary projectors, is to delete the following elements from A_w, B_w, C_w, D_w, for $w = u, v$ and for all $i \in [M]$ (note that these elements may have not been present to begin with, or may be "removed" multiple times between the two operations—this does not cause any issues):

  - $d_\alpha + (i-1)d_{\alpha+1}$ from A_w.
  - $id_{\alpha+1}$ from B_w.
  - $d_\beta + (i-1)d_{\alpha+1}$ from C_w.
  - $id_{\beta+1}$ from D_w.

We can now describe the algorithm.

Quantum Stage For each $\overrightarrow{uv}$ processed until the quantum stage terminates:

1. `inc`$(\mathcal{E}, w, 1)$ for $w = u, v$, and $\mathcal{E} = \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$.
2. If $f_\alpha(\overrightarrow{uv}) = 1$, `inc`$(\mathcal{A}, u, d_{\alpha+1})$ and `inc`$(\mathcal{B}, u, d_{\alpha+1})$.
3. If $f_\beta(\overrightarrow{uv}) = 1$, `inc`$(\mathcal{C}, u, d_{\beta+1})$ and `inc`$(\mathcal{D}, u, d_{\beta+1})$.
4. `measure`(u, v). If the measurement returns $|\nu^r_{i,j,b}\rangle$, pass it along with u, v to the classical stage and continue.
5. `cleanup`(u, v). If the measurement returns anything other than the projector onto the complement of the cleanup vectors, immediately terminate the algorithm, outputting an all-zeroes estimate.

If the quantum stage processes every edge without being terminated by a measurement outcome, output an all-zeroes estimate and skip the classical stage.

Classical Stage For the remainder of the stream, track $d_u^{\text{out},>e}$,$d_v^{\text{out},>e}$, $d_u^{>e}$,$d_v^{>e}$ (giving us exact values for these variables). Then estimate $d_u^{\leq e}$, $d_v^{\leq e}$ by assuming that they are equal to d_α, d_β, respectively. Then estimate $\widetilde{d}_u^{\text{out},\leq e}$, $\widetilde{d}_v^{\text{out},\leq e}$, by assuming that the number of edges e with head u and $f_\alpha(e) = 1$ is $i-1$, and the number with head v and $f_\beta(e) = 1$ is $j-1$.

Combine these estimates and evaluate $g(u)$, $g(v)$ to estimate $\widetilde{b}_u^e$ and $\widetilde{b}_v^e$. If $r = 1$ or 4, set $(-1)^b M/2$ as the corresponding entry of the pseudosnapshot estimate (with every other entry as 0). If $r = 2$ or 3, set it as $(-1)^{\bar{b}} M/2$ instead.

6.2 Analysis

6.2.1 State Invariant

Lemma 11. *Consider any time after some number of edges have been (completely) processed in the quantum stage, and suppose the state has not yet terminated, and t has not exceeded M. Let $v \in V$, and let r be the number of those edges that were incident to v. Let R be the number of those edges e such that v was the head of the edge, and $f_\alpha(e) = 1$.*

Then, if $R = 0$,

$$A_v = \mathbb{N} \cap [1, \min(r+1, d_\alpha))$$
$$B_v = \mathbb{N} \cap [1, \min(r+1, d_{\alpha+1}))$$

and if $R > 0$, there exists $(\rho_i)_{i=1}^R \in [r]^R$ such that:

$$A_v = \mathbb{N} \cap \left([1, d_\alpha) \cup \bigcup_{i=1}^{R} I_i \right)$$
$$B_v = \mathbb{N} \cap \left([1, d_{\alpha+1}) \cup \bigcup_{i=1}^{R} J_i \right)$$

Where

$$I_i = \begin{cases} [d_\alpha + (i-1)d_{\alpha+1} + \rho_i, d_\alpha + id_{\alpha+1}) & i < R \\ [d_\alpha + (R-1)d_{\alpha+1} + \rho_R, Rd_{\alpha+1} + \min(r+1, d_\alpha)) & i = R \end{cases}$$
$$J_i = \begin{cases} [id_{\alpha+1} + \rho_i, (i+1)d_{\alpha+1}) & i < R \\ [Rd_{\alpha+1} + \rho_R, Rd_{\alpha+1} + \min(r+1, d_{\alpha+1})) & i = R \end{cases}$$

The same relationship holds for C_v and D_v, except with β instead of α.

Proof. We will prove the result for A_v and B_v. The proof for C_v and D_v is identical, with β substituted for α. Note that when describing the updates performed on seeing an edge $\overrightarrow{uv}$ or $\overrightarrow{vu}$ we will ignore the updates that only touch u, as they have no effect on the sets we are analyzing here.

We prove this result by induction. First, we consider $R = 0$ and $r = 0$ as the base case. Next, we prove the inductive step where r is increased by 1 while $R = 0$. This finishes the proof for $R = 0$

and any r. Next, we prove the result for $R = 1$ by considering the step when R is increased from $R = 0$. Note that any update that increases R also increases r, so in this case r is also increased by 1. Last, we fix any r and any $R > 0$ and prove the inductive step when r is increased by 1, but R stays unchanged, and when both r and R are increased by 1, which completes the proof.

We start with the case where $R = 0$ and $r = 0$. Then the statement is equivalent to $A_v = B_v = \emptyset$, which follows from how we have defined the initial state of the algorithm, and the fact that edges not incident to v do not result in updates that affect A_v or B_v.

Now suppose the result holds for r (with R still 0), and consider the $(r+1)^{\text{th}}$ edge incident to v to arrive (with no $\overrightarrow{vw}$ such that $f(\overrightarrow{vw}) = 1$ having arrived yet). Then the update consists of performing $\texttt{inc}(\mathcal{A}, v, 1)$, $\texttt{inc}(\mathcal{B}, v, 1)$, and then $\texttt{measure}(v, w)$ or $\texttt{measure}(w, v)$ for some w, followed by $\texttt{cleanup}(v, w)$ or $\texttt{cleanup}(w, v)$. Before the measurement, this gives us

$$\begin{aligned} A_v &= \mathbb{N} \cap [1, \min(r+2, d_\alpha + 1)) \\ B_v &= \mathbb{N} \cap [1, \min(r+2, d_{\alpha+1} + 1))) \end{aligned}$$

and as the condition of the lemma supposes that the measurement did not result in the algorithm terminating, it will have removed d_α from A_v and $d_{\alpha+1}$ from B_v, and so

$$\begin{aligned} A_v &= \mathbb{N} \cap [1, \min(r+2, d_\alpha)) \\ B_v &= \mathbb{N} \cap [1, \min(r+2, d_{\alpha+1}))) \end{aligned}$$

as desired.

Next we will prove the result for R increased from 0 to 1 and r increased by 1. By the previous section, when $R = 0$ the state is

$$\begin{aligned} A_v &= \mathbb{N} \cap [1, \min(r+1, d_\alpha)) \\ B_v &= \mathbb{N} \cap [1, \min(r+1, d_{\alpha+1}))). \end{aligned}$$

When the $(r+1)^{\text{th}}$ edge incident to v arrives, if it is also the first edge with head v in $f_\alpha^{-1}(1)$, the update consists of performing $\texttt{inc}(\mathcal{A}, v, 1)$, $\texttt{inc}(\mathcal{B}, v, 1)$, $\texttt{inc}(\mathcal{A}, v, d_{\alpha+1})$, $\texttt{inc}(\mathcal{B}, v, d_{\alpha+1})$, and then $\texttt{measure}(v, w)$ for some w, followed by $\texttt{cleanup}(v, w)$.

After the `inc` operations are performed we have

$$\begin{aligned} A_v &= \mathbb{N} \cap [1, \min(r + d_{\alpha+1} + 2, d_\alpha + d_{\alpha+1} + 1)) \\ B_v &= \mathbb{N} \cap [1, \min(r + d_{\alpha+1} + 2, 2d_{\alpha+1} + 1))) \end{aligned}$$

and then after performing the measurements we remove d_α and $d_\alpha + d_{\alpha+1}$ from A_v and $d_{\alpha+1}$ and $2d_{\alpha+1}$ from B_v, so we have

$$\begin{aligned} A_v &= \mathbb{N} \cap ([1, d_\alpha) \cup [d_\alpha + 1, \min(r + d_{\alpha+1} + 2, d_\alpha + d_{\alpha+1}))) \\ B_v &= \mathbb{N} \cap ([1, d_{\alpha+1}) \cup [d_{\alpha+1} + 1, \min(r + d_{\alpha+1} + 2, 2d_{\alpha+1})))) \end{aligned}$$

which matches the lemma statement by setting $\rho_1 = \rho_R = 1$.

We now need to consider two more cases to complete the proof: First, the case when we have the result for some $R > 0$ and r, and the update consists of the $(r+1)^{\text{th}}$ edge incident to v, and this edge is *not* in $f_\alpha^{-1}(1)$ or does not have v as its head. Second, the case when we have the result for R and r and the update consists of the $(r+1)^{\text{th}}$ edge incident to v, which is also the $(R+1)^{\text{th}}$ edge with head v in $f_\alpha^{-1}(1)$.

Let us deal with the cases when r is increased from r to $r+1$ and R stays the same first. The update consists of performing $\texttt{inc}(\mathcal{A}, v, 1)$, $\texttt{inc}(\mathcal{B}, v, 1)$, and then $\texttt{measure}(v, w)$ or $\texttt{measure}(w, v)$ for some w, followed by $\texttt{cleanup}(v, w)$ or $\texttt{cleanup}(w, v)$. The state before the update, by the inductive hypothesis, is

$$A_v = \mathbb{N} \cap \left([1, d_\alpha) \cup \bigcup_{i=1}^{R} I_i \right)$$
$$B_v = \mathbb{N} \cap \left([1, d_{\alpha+1}) \cup \bigcup_{i=1}^{R} J_i \right)$$

with the intervals I_i and J_i as defined in the lemma statement. The $\texttt{inc}(\mathcal{A}, v, 1)$ and $\texttt{inc}(\mathcal{B}, v, 1)$ operations, then, correspond to replacing these intervals with

$$I_i' = \begin{cases} [d_\alpha + (i-1)d_{\alpha+1} + \rho_i + 1, d_\alpha + id_{\alpha+1} + 1) & \text{i} < \text{R} \\ [d_\alpha + (R-1)d_{\alpha+1} + \rho_R + 1, Rd_{\alpha+1} + \min(r+2, d_\alpha + 1)) & \text{i} = \text{R} \end{cases}$$
$$J_i' = \begin{cases} [id_{\alpha+1} + \rho_i + 1, (i+1)d_{\alpha+1} + 1) & \text{i} < \text{R} \\ [Rd_{\alpha+1} + \rho_R + 1, Rd_{\alpha+1} + \min(r+2, d_{\alpha+1} + 1)) & \text{i} = \text{R} \end{cases}$$

and $[1, d_\alpha)$ with $[1, d_\alpha + 1)$, and $[1, d_{\alpha+1})$ with $[1, d_{\alpha+1} + 1)$.

The measurements then delete $d_\alpha + (i-1)d_\alpha$ from A_v and id_α from B_v for every $i \in [M]$. So $[1, d_\alpha + 1)$ and $[1, d_{\alpha+1})$ return to $[1, d_\alpha)$, $[1, d_{\alpha+1})$, respectively, and the intervals I_i, J_i become

$$I_i'' = \begin{cases} [d_\alpha + (i-1)d_{\alpha+1} + \rho_i + 1, d_\alpha + id_{\alpha+1}) & \text{i} < \text{R} \\ [d_\alpha + (R-1)d_{\alpha+1} + \rho_R + 1, Rd_{\alpha+1} + \min(r+2, d_\alpha)) & \text{i} = \text{R} \end{cases}$$
$$J_i'' = \begin{cases} [id_{\alpha+1} + \rho_i + 1, (i+1)d_\alpha) & \text{i} < \text{R} \\ [Rd_{\alpha+1} + \rho_R + 1, Rd_{\alpha+1} + \min(r+2, d_{\alpha+1})) & \text{i} = \text{R} \end{cases}$$

and so by setting the new I_i to be I_i'' and likewise with J_i, the states are in the form desired (by incrementing every element of $(\rho_i)_{i=1}^R$ by 1), as r is now 1 larger.

Finally, we consider the inductive step where both R and r are increased by 1. This means the arriving edge is the $(r+1)^{\text{th}}$ edge incident to v, which is also the $(R+1)^{\text{th}}$ edge with head v in $f_\alpha^{-1}(1)$.

The update consists of performing $\texttt{inc}(\mathcal{A}, v, 1)$, $\texttt{inc}(\mathcal{B}, v, 1)$, $\texttt{inc}(\mathcal{A}, v, d_{\alpha+1})$, $\texttt{inc}(\mathcal{B}, v, d_{\alpha+1})$, and then $\texttt{measure}(v, w)$ for some w, followed by $\texttt{cleanup}(v, w)$. The state before the update, by the inductive hypothesis, is

$$A_v = \mathbb{N} \cap \left([1, d_\alpha) \cup \bigcup_{i=1}^{R} I_i \right)$$
$$B_v = \mathbb{N} \cap \left([1, d_{\alpha+1}) \cup \bigcup_{i=1}^{R} J_i \right)$$

with the intervals I_i and J_i as defined in the lemma statement. As $\texttt{inc}$ operations add together, after all the increment operations we have replaced $[1, d_\alpha)$ with $[1, d_\alpha + d_{\alpha+1} + 1)$ and $[1, d_{\alpha+1})$ with

$[1, 2d_{\alpha+1}+1)$ and I_i, J_i with

$$I'_i = \begin{cases} [d_\alpha + id_{\alpha+1} + \rho_i + 1, d_\alpha + (i+1)d_{\alpha+1} + 1) & \text{i} < \text{R} \\ [d_\alpha + Rd_{\alpha+1} + \rho_R + 1, (R+1)d_{\alpha+1} + \min(r+2, d_\alpha + 1)) & \text{i} = \text{R} \end{cases}$$

$$J'_i = \begin{cases} [(i+1)d_{\alpha+1} + \rho_i + 1, (i+2)d_{\alpha+1} + 1) & \text{i} < \text{R} \\ [(R+1)d_{\alpha+1} + \rho_R + 1, (R+1)d_{\alpha+1} + \min(r+2, d_{\alpha+1} + 1)) & \text{i} = \text{R} \end{cases}$$

and so after the measurements, as they remove $d_\alpha + (i-1)d_{\alpha+1}$ from A_v and $id_{\alpha+1}$ from B_v for all i, $[1, d_\alpha + d_{\alpha+1} + 1)$ becomes $[1, d_\alpha) \cup [d_\alpha + 1, d_\alpha + d_{\alpha+1})$ and $[1, 2d_{\alpha+1} + 1)$ becomes $[1, d_{\alpha+1}) \cup [d_{\alpha+1} + 1, 2d_{\alpha+1})$. I'_i and J'_i become

$$I''_i = \begin{cases} [d_\alpha + id_{\alpha+1} + \rho_i + 1, d_\alpha + (i+1)d_{\alpha+1}) & \text{i} < \text{R} \\ [d_\alpha + Rd_{\alpha+1} + \rho_R + 1, (R+1)d_{\alpha+1} + \min(r+2, d_\alpha)) & \text{i} = \text{R} \end{cases}$$

$$J''_i = \begin{cases} [(i+1)d_{\alpha+1} + \rho_i + 1, (i+2)d_{\alpha+1}) & \text{i} < \text{R} \\ [(R+1)d_{\alpha+1} + \rho_R + 1, (R+1)d_{\alpha+1} + \min(r+2, d_{\alpha+1})) & \text{i} = \text{R} \end{cases}$$

and so the state is of the form required, by setting (writing ρ'_i for the old ρ_i, and noting that this means $\rho_i \leq r+1$ for all i)

$$I_i = \begin{cases} [d_\alpha + 1, d_\alpha + d_{\alpha+1}) & i = 1 \\ I''_{i-1} & 1 < i \leq R+1 \end{cases}$$

$$J_i = \begin{cases} [d_{\alpha+1} + 1, 2d_{\alpha+1}) & i = 1 \\ J''_{i-1} & 1 < i \leq R+1 \end{cases}$$

$$\rho_i = \begin{cases} 1 & \text{i} = 1 \\ \rho'_{i-1} + 1 & 1 < i \leq R+1 \end{cases}$$

which completes the proof. □

6.2.2 Measurement Outcomes

In this section we characterize the expected effect on the pseudosnapshot estimate from all of the measurement outcomes that can terminate the quantum stage of the algorithm: those other than the residual projector in `measure` and `cleanup`.

Lemma 12. *For all u, v, the expected contribution of* `cleanup`(u, v) *to every entry of the pseudosnapshot estimate is* 0*.*

Proof. This follows immediately from the fact that we do not modify the estimate when the algorithm terminates due to `cleanup`. □

In order to analyze the expectation of the estimate output by the algorithm, we will need the following lemma about the probability of the algorithm terminating before a certain point. Note that for any stream of edges, the sequence of measurements performed by the algorithm is deterministic, except that the sequence may be terminated early depending on the measurement results.

Lemma 13. *For any $k \in \mathbb{N}$, let p_k be the probability that the algorithm terminates in the first k measurements performed by the algorithm. Then, if the algorithm does not terminate, the value of M' after these measurements is $(1-p_k)M$.*

Proof. We will prove a slightly stronger version of the result, in which we treat `measure` as $4\kappa^2$ sequential measurements on

$$|\nu^a_{i,j,0}\rangle\langle\nu^a_{i,j,0}|, |\nu^a_{i,j,1}\rangle\langle\nu^a_{i,j,1}|, \mathbb{1} - |\nu^a_{i,j,0}\rangle\langle\nu^a_{i,j,0}| - |\nu^a_{i,j,1}\rangle\langle\nu^a_{i,j,1}|$$

for $a \in [4]$, $(i,j) \in [\kappa]^2$, and `cleanup` as $16M\kappa^2$ sequential measurements on

$$P, \mathbb{1} - P$$

for each of the projectors P used in `cleanup`. The lemma result is then just a special case in which k is restricted to those values corresponding to a discrete set of `measure` and `cleanup` measurements.

We proceed by induction on k. For $k = 0$, $p_k = 0$ and $M' = M$. So suppose that after k measurements, $M' = (1-p_k)M$. Then either the next measurement is from a `measure` or `cleanup` operation. Suppose it is from a `measure` operation. Then

$$|\nu^a_{i,j,b}\rangle = \frac{|x\rangle + (-1)^b|y\rangle}{\sqrt{2}}$$

for two states $|x\rangle$, $|y\rangle$ that may or may not be in the M'-element superposition held by the algorithm. Call this superposition $|\phi\rangle$. Then

$$|\langle\phi|\nu^a_{i,j,b}\rangle|^2 = \begin{cases} \frac{(1+(-1)^b)^2}{2M'} = \frac{(1+(-1)^b)^2}{2(1-p_k)M} & \text{if both states are in the superposition} \\ \frac{1}{2M'} = \frac{1}{2(1-p_k)M} & \text{if one of them is} \\ 0 & \text{otherwise.} \end{cases}$$

So in the first of these cases, the probability that one of $|\nu^a_{i,j,b}\rangle_{b=1,2}$ is returned, terminating the algorithm, is $2/M'$. So $1-p_{k+1} = (1-2/(1-p_k)M)(1-p_k) = 1-p_k-2/M$. If the algorithm does not terminate, both states are removed from the superposition and M' becomes $(1-p_k)M - 2 = (1-p_{k+1})M$.

In the second, the probability of termination is $2 \cdot 1/2(1-p_k)M = 1/(1-p_k)M$, and so $1-p_{k+1} = (1-1/(1-p_k)M)(1-p_k) = 1-p_k-1/M$. If the algorithm does not terminate, one state is removed from the superposition and M' becomes $(1-p_k)M - 1 = (1-p_{k+1})M$.

In the third, $p_{k+1} = p_k$ and the superposition is unchanged, so M' remains $(1-p_k)M = (1-p_{k+1})M$.

Now suppose the measurement is from a `cleanup` operation. Then the measurement uses a projector onto a state $|x\rangle$ that may or may not be in the superposition. If it is, the probability of termination is $1/(1-p_k)M$ and M' becomes $(1-p_k)M - 1$ if the algorithm does not terminate, so $M' = (1-p_{k+1})M$. If it is not, $p_{k+1} = p_k$ and M' is unchanged, so M' remains $(1-p_{k+1})M$. □

Lemma 14. *For every $\overrightarrow{uv} \in E$ such that $d_u^{\leq \overrightarrow{uv}} \in [d_\alpha, d_{\alpha+1})$, $d_v^{\leq \overrightarrow{uv}} \in [d_\beta, d_{\beta+1})$, and for every $(i,j) \in [\kappa]^2$, the total expected contribution of `measure`(u,v) to the pseudosnapshot estimate from returning one of $|\nu^k_{i,j,b}\rangle_{k\in[4],b\in\{0,1\}}$ is 1 to the $\widetilde{b}_u^{\overrightarrow{uv}}$, $\widetilde{b}_v^{\overrightarrow{uv}}$ entry and zero to all other entries if $i = \frac{\kappa}{2d_\alpha}\widetilde{d}_u^{out,\leq\overrightarrow{uv}} + 1$ and $j = \frac{\kappa}{2d_\beta}\widetilde{d}_v^{out,\leq\overrightarrow{uv}} + 1$. Otherwise it is zero everywhere.*

Proof. We will start by analyzing the expectation conditional on the algorithm not terminating before $\overrightarrow{uv}$ arrives. This will give us the result in terms of M'. We will then use Lemma 13 to give us the expectation in terms of M.

At the time when $\overrightarrow{uv}$ arrives, let R'_u, R'_v, be the R in Lemma 11 for u, v respectively, and likewise for r'_u, r'_v and r. Then, for $w = u, v$, set $r_w = r'_w + 1$. Set $R_u = R'_u + f_\alpha(\overrightarrow{uv}) + f_\beta(\overrightarrow{uv})$. Note that this means that, before measuring, the algorithm will update the state with an $\texttt{inc}(\mathcal{E}, w, 1)$ operation for $w = u, v$ and $\mathcal{E} = \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$, an $\texttt{inc}(\mathcal{E}, u, d_{\alpha+1} \cdot f_\alpha(\overrightarrow{uv}))$ operation for $\mathcal{E} = \mathcal{A}, \mathcal{B}$ and an $\texttt{inc}(\mathcal{E}, u, d_{\beta+1} \cdot f_\beta(\overrightarrow{uv}))$ operation for $\mathcal{E} = \mathcal{C}, \mathcal{D}$.

Note that $R_u = \frac{\kappa}{2d_\alpha}\widetilde{d}_u^{\text{out},\leq\overrightarrow{uv}}$, $R_v = \frac{\kappa}{2d_\beta}\widetilde{d}_v^{\text{out},\leq\overrightarrow{uv}}$, $r_u = d_u^{\leq\overrightarrow{uv}}$, $r_v = d_v^{\leq\overrightarrow{uv}}$. So we have $r_u \in [d_\alpha, d_{\alpha+1})$, $r_v \in [d_\beta, d_{\beta+1})$.

By Lemma 11, this means that for all i, before the increments,

$$d_\alpha + id_{\alpha+1} - 1 \in A_u, id_{\alpha+1} - 1 \in B_u \text{ iff } i \in [R'_u]$$
$$d_\beta + id_{\beta+1} - 1 \in C_v, id_{\beta+1} - 1 \in D_v \text{ iff } i \in [R'_v]$$

and after them:

$$d_\alpha + id_{\alpha+1} \in A_u, id_{\alpha+1} \in B_u \text{ iff } i \in [R_u]$$
$$d_\beta + id_{\beta+1} \in C_v, id_{\beta+1} \in D_v \text{ iff } i \in [R_v]$$

Now, writing $|\phi\rangle$ for the state of the algorithm before the measurements, and recalling that

$$|\nu^1_{i,j,b}\rangle = \frac{|u(d_\alpha + (i-1)d_{\alpha+1})a^{r_{i,j}}\rangle + (-1)^b|v(d_\beta + (j-1)d_{\beta+1})c^{r_{i,j}}\rangle}{\sqrt{2}}$$
$$|\nu^2_{i,j,b}\rangle = \frac{|u(id_{\alpha+1})b^{r_{i,j}}\rangle + (-1)^b|v(d_\beta + (j-1)d_{\beta+1})c^{s_{i,j}}\rangle}{\sqrt{2}}$$
$$|\nu^3_{i,j,b}\rangle = \frac{|u(d_\alpha + (i-1)d_{\alpha+1})a^{s_{i,j}}\rangle + (-1)^b|v(jd_{\beta+1})d^{r_{i,j}}\rangle}{\sqrt{2}}$$
$$|\nu^4_{i,j,b}\rangle = \frac{|u(id_{\alpha+1})b^{s_{i,j}}\rangle + (-1)^b|v(jd_{\beta+1})d^{s_{i,j}}\rangle}{\sqrt{2}}$$

we have

$$|\langle\phi|\nu^1_{i,j,b}\rangle|^2 = \begin{cases} \frac{\left(1+(-1)^b\right)^2}{2M'} & i \in [R_u + 1] \text{ and } j \in [R_v + 1] \\ \frac{1}{2M'} & \text{exactly one of } i \in [R_u + 1] \text{ and } j \in [R_v + 1] \\ 0 & \text{otherwise} \end{cases}$$

$$|\langle\phi|\nu^2_{i,j,b}\rangle|^2 = \begin{cases} \frac{\left(1+(-1)^b\right)^2}{2M'} & i \in [R_u] \text{ and } j \in [R_v + 1] \\ \frac{1}{2M'} & \text{exactly one of } i \in [R_u] \text{ and } j \in [R_v + 1] \\ 0 & \text{otherwise} \end{cases}$$

$$|\langle\phi|\nu^3_{i,j,b}\rangle|^2 = \begin{cases} \frac{\left(1+(-1)^b\right)^2}{2M'} & i \in [R_u + 1] \text{ and } j \in [R_v] \\ \frac{1}{2M'} & \text{exactly one of } i \in [R_u + 1] \text{ and } j \in [R_v] \\ 0 & \text{otherwise} \end{cases}$$

$$|\langle\phi|\nu^4_{i,j,b}\rangle|^2 = \begin{cases} \frac{\left(1+(-1)^b\right)^2}{2M'} & i \in [R_u] \text{ and } j \in [R_v] \\ \frac{1}{2M'} & \text{exactly one of } i \in [R_u] \text{ and } j \in [R_v] \\ 0 & \text{otherwise.} \end{cases}$$

Now, recall that the contribution to the chosen entry of the pseudosnapshot (with the choice depending only on i, j) when seeing the result $|\nu^a_{i,j,b}\rangle$ is $(-1)^b M/2$ if $a = 1, 4$ and $(-1)^{\bar{b}} M/2$ if $a = 2, 3$. Therefore, writing the total expected contribution from $|\nu^a_{i,j,b}\rangle$ summed over $b = 0, 1$ as $x^a_{i,j}$, we have

$$x^1_{i,j} = \begin{cases} \frac{M}{M'} & i \in [R_u + 1] \text{ and } j \in [R_v + 1] \\ 0 & \text{otherwise} \end{cases}$$

$$x^2_{i,j} = \begin{cases} -\frac{M}{M'} & i \in [R_u] \text{ and } j \in [R_v + 1] \\ 0 & \text{otherwise} \end{cases}$$

$$x^3_{i,j} = \begin{cases} -\frac{M}{M'} & i \in [R_u + 1] \text{ and } j \in [R_v] \\ 0 & \text{otherwise} \end{cases}$$

$$x^4_{i,j} = \begin{cases} \frac{M}{M'} & i \in [R_u] \text{ and } j \in [R_v] \\ 0 & \text{otherwise} \end{cases}$$

and so

$$\sum_{a=1}^{4} x^a_{i,j} = \begin{cases} \frac{M}{M'} & i = R_u + 1 \text{ and } j = R_v + 1 \\ 0 & \text{otherwise.} \end{cases}$$

Now, when $i = R_u + 1$ and $j = R_v + 1$, and $d_u^{\leq \overrightarrow{uv}} \in [d_\alpha, d_{\alpha+1})$, $d_{\bar{v}}^{\leq \overrightarrow{uv}} \in [d_\beta, d_{\beta+1})$, the entry of the pseudosnapshot estimate chosen is $\widetilde{b}_u^{\overrightarrow{uv}}, \widetilde{b}_v^{\overrightarrow{uv}}$. We have that the contribution to it conditioned on the algorithm not terminating before processing $\overrightarrow{uv}$ is M/M'. So, as the contribution from $\overrightarrow{uv}$ is guaranteed to be 0 if the algorithm has already terminated, the total expectation is, by Lemma 13,

$$(1 - p) \cdot \frac{M}{M'} = (1 - p) \cdot \frac{M}{(1 - p)M} = 1$$

where p is the probability of termination before processing $\overrightarrow{uv}$. □

Lemma 15. *For every $\overrightarrow{uv}$ such that $d_u^{\leq \overrightarrow{uv}} \notin [d_\alpha, d_{\alpha+1})$, or $d_{\bar{v}}^{\leq \overrightarrow{uv}} \notin [d_\beta, d_{\beta+1})$, the total expected contribution of* `measure`(u, v) *to any entry of the pseudosnapshot estimate is zero.*

Proof. We will analyze the expectation conditional on the algorithm not terminating before $\overrightarrow{uv}$ arrives. As the expected contribution is trivially 0 conditional on the algorithm terminating before $\overrightarrow{uv}$ arrives, this will suffice for the result.

We have that at least one of $d_{\bar{w}}^{\leq \overrightarrow{uv}} < d_\gamma$ for some $(w, \gamma) \in \{(u, \alpha), (v, \beta)\}$ or $d_{\bar{w}}^{\leq \overrightarrow{uv}} \geq d_{\gamma+1}$ for some $(w, \gamma) \in \{(u, \alpha), (v, \beta)\}$. By Lemma 11, this implies that, for one of $(E, F) \in \{(A, B), (C, D)\}$,

$$d_\gamma + (i - 1)d_{\gamma+1} \in E_w \Leftrightarrow i d_{\gamma+1} \in F_w$$

for all $i \in [M]$. So, writing $|\phi\rangle$ for the state of the algorithm, we have either

$$|\langle\phi|\nu^1_{i,j,b}\rangle| = |\langle\phi|\nu^2_{i,j,b}\rangle|, |\langle\phi|\nu^3_{i,j,b}\rangle| = |\langle\phi|\nu^4_{i,j,b}\rangle|$$

for all i, j, b, or

$$|\langle\phi|\nu^1_{i,j,b}\rangle| = |\langle\phi|\nu^3_{i,j,b}\rangle|, |\langle\phi|\nu^2_{i,j,b}\rangle| = |\langle\phi|\nu^4_{i,j,b}\rangle|$$

for all i, j, b. As for all i, j, b the contribution from the measurement result $|\nu^a_{i,j,b}\rangle$ is made to the same entry of the estimate regardless of a, and is $(-1)^b M/2$ for $a = 1, 4$ and $(-1)^{\bar{b}} M/2$ for $a = 2, 3$, this implies the expected contributions cancel out, and so the lemma follows. □

Lemma 16. *Each entry of the estimate has bias at most the number of edges $\overrightarrow{uv}$ such that:*

1. $d_u^{\leq \overrightarrow{uv}} \in [d_\alpha, d_{\alpha+1})$
2. $d_v^{\leq \overrightarrow{uv}} \in [d_\beta, d_{\beta+1})$
3. $\max\left\{ \frac{\kappa}{2d_\alpha} \widetilde{d}_u^{out,\leq \overrightarrow{uv}} + 1, \frac{\kappa}{2d_\beta} \widetilde{d}_v^{out,\leq \overrightarrow{uv}} + 1 \right\} > \kappa$

Proof. By Lemma 15, no edge edge with endpoints in the wrong degree classes contribute to the estimate. By Lemma 14 every edge $\overrightarrow{uv}$ with endpoints in the right degree classes contributes 1 to the correct entry of the estimate of PsSnap^G (and 0 to all others), unless

$$\max\left\{ \frac{\kappa}{2d_\alpha} \widetilde{d}_u^{\mathrm{out},\leq \overrightarrow{uv}} + 1, \frac{\kappa}{2d_\beta} \widetilde{d}_v^{\mathrm{out},\leq \overrightarrow{uv}} + 1 \right\} > \kappa$$

in which case it contributes 0 to all entries, as the i, j in Lemma 14 only range in $[\kappa]^2$. □

Lemma 17. *The variance of any entry of the pseudosnapshot estimate is* $\mathrm{O}\big(\kappa^6 m^2\big)$.

Proof. The output of the algorithm is an estimate of the pseudosnapshot with one $\mathrm{O}(M)$ entry and all other entries 0. So this follows by the fact that $M = \mathrm{O}\big(\kappa^3 m\big)$. □

6.2.3 Space Usage

Lemma 18. *The algorithm uses only* $\mathrm{O}(\log n)$ *qubits of space.*

Proof. The algorithm maintains a single state that requires $\mathrm{O}(\log M) = \mathrm{O}(\log n)$ qubits to store, along with a constant number of counters of size $\mathrm{poly}(n)$, along with some rational numbers made from constant-degree polynomials of such numbers. □

7 Quantum Algorithm for Max-DiCut

We may now prove our main result.

Theorem 19. *There is a quantum streaming algorithm which* 0.4844*-approximates the* Max-DiCut *value of an input graph G with probability $1 - \delta$. The algorithm uses* $\mathrm{O}\big(\log^5 n \log \frac{1}{\delta}\big)$ *qubits of space.*

Proof. Now, by Lemma 9, there is an $\alpha > 0.4844$, vector $\mathbf{r}$ and constant $C > 0$ such that

$$\mathbf{r}^\dagger \mathsf{PsSnap}^G (1 - \mathbf{r}) - C\varepsilon m$$

gives an $\alpha' = \alpha - \mathrm{O}(\varepsilon)$-approximation of the Max-DiCut value of G with probability $1 - \mathrm{O}(\varepsilon) - e^{\mathrm{O}\left(\varepsilon^6 \kappa\right)}$.

Recall that PsSnap^G depends on the accuracy parameters κ and ε we choose. We will choose ε to be a small enough constant that $\alpha' > 0.4844$ and the $1 - \mathrm{O}(\varepsilon)$ term in the success probability is at least $4/5$. κ will be chosen to depend only on ε. We will choose it such that the $1 - \mathrm{O}(\varepsilon) - e^{\mathrm{O}\left(\varepsilon^6 \kappa\right)}$ success probability is at least $3/4$.

Now we just need to approximate PsSnap^G. By applying Lemma 10 $\mathrm{O}\big(\log^2 n\big)$ times in parallel (once for each pair of degree classes $[d_i, d_{i+1})$) and summing the outputs, we get a $\mathrm{O}\big(\log^3 n\big)$ space (as ε and therefore κ will be chosen to be constant) algorithm that outputs an estimate of PsSnap^G where every entry has variance $\mathrm{O}\big(\kappa^6 m^2 \log^2 n\big)$, and bias at most the number of edges $\overrightarrow{uv}$ such that

$$\max\left\{\frac{\kappa}{2d_\alpha}\widetilde{d}_u^{\mathrm{out},\leq\overrightarrow{uv}}+1, \frac{\kappa}{2d_\beta}\widetilde{d}_v^{\mathrm{out},\leq\overrightarrow{uv}}+1\right\} > \kappa$$

where α, β are the unique indices in $\{0,\dots,\lfloor\log_{1+\varepsilon^3} n\rfloor\}$ such that $u \in [d_\alpha, d_{\alpha+1})$ and $v \in [d_\beta, d_{\beta+1})$. Now for each of $(w,\gamma) = (u,\alpha),(v,\beta)$, $\frac{\kappa}{2d_\gamma}\widetilde{d}_w^{\mathrm{out},\leq\overrightarrow{uv}}$ is distributed as $\mathrm{B}(d_w^{\leq\overrightarrow{uv}}, \kappa/2d_\gamma)$, and so as $d_w^{\leq\overrightarrow{uv}} \leq (1+\varepsilon^3)d_\gamma$, the probability that it is at least $\kappa-1$ is $2^{-\Omega(\kappa)}$. So the expectation of the total bias is at most $2^{-\Omega(\kappa)}m$.

So by Markov's inequality, this bias is at most $2^{-\Omega(\kappa)}m$ with probability at most $1-\varepsilon$ over f if we choose κ (as a function of ε) small enough.

We may repeat this process $\Theta\big(\kappa^6 \frac{1}{\varepsilon^3}\log^2 n\big)$ times in parallel (with the same hash functions each time), at $\mathrm{O}\big(\log^5 n\big)$ space cost, and pointwise average the snapshot estimates in order to obtain a $\mathrm{O}\big(\varepsilon^3 m\big)$ variance estimate of PsSnap^G. By Chebyshev's inequality, with probability $1-\varepsilon$, every entry of this estimate is within εm of the corresponding entry of PsSnap^G. Therefore, calling this estimate M,

$$\mathbf{r}^\dagger M(1-\mathbf{r}) - D\varepsilon m$$

gives an $(\alpha' - \mathrm{O}(\varepsilon))$-approximation of the Max-DiCut value of G with probability $3/4 - \mathrm{O}(\varepsilon)$. So choosing ε to be a small enough constant, this gives us a 0.4844-approximation with probability $2/3$.

The result then follows by running the entire procedure $\Theta\big(\log\frac{1}{\delta}\big)$ times in parallel (sampling new hash functions each time), and taking the median of the Max-DiCut(G) estimates. □

Acknowledgements

Nadezhda Voronova thanks Mark Bun for many helpful conversations. Her research was supported by NSF award CNS-2046425.

John Kallaugher and Ojas Parekh were supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. Also supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing program, Fundamental Algorithmic Research for Quantum Computing (FAR-QC). Part of this research was performed while Ojas Parekh was visiting the Institute for Pure and Applied Mathematics (IPAM), which is supported by the National Science Foundation (Grant No. DMS-1925919).

This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article

References

[AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 20–29, New York, NY, USA, 1996. Association for Computing Machinery.

[BHPZ22] Joshua Brakensiek, Neng Huang, Aaron Potechin, and Uri Zwick. Separating max 2-and, max di-cut and max cut. *arXiv preprint arXiv:2212.11191*, 2022.

[BKS02] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 623–632, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[BOV13] Vladimir Braverman, Rafail Ostrovsky, and Dan Vilenchik. How hard is counting triangles in the streaming model? In *Automata, Languages, and Programming*, pages 244–254. Springer, 2013.

[CGV20] Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal streaming approximations for all boolean Max-2CSPs and Max-kSAT. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 330–341. IEEE, 2020.

[FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[FJ15] Uriel Feige and Shlomo Jozeph. Oblivious algorithms for the maximum directed cut problem. *Algorithmica*, 71:409–428, 2015.

[Fla85] Philippe Flajolet. Approximate counting: a detailed analysis. *BIT Numerical Mathematics*, 25(1):113–134, 1985.

[FN85] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[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 J. Comput.*, 38(5):1695–1708, 2008.

[JK21] Rajesh Jayaram and John Kallaugher. An optimal algorithm for triangle counting in the stream. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 11:1–11:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[JN14] Rahul Jain and Ashwin Nayak. The Space Complexity of Recognizing Well-Parenthesized Expressions in the Streaming Model: The Index Function Revisited. *IEEE Transactions on Information Theory*, 60(10):6646–6668, October 2014.

[Kal22] John Kallaugher. A quantum advantage for a natural streaming problem. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 897–908, 2022.

[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, pages 277–288, New York, NY, USA, 2019. Association for Computing Machinery.

[KP22] John Kallaugher and Ojas Parekh. The quantum and classical streaming complexity of quantum and classical Max-Cut. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 498–506. IEEE, 2022.

[LLZ02] Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the max 2-sat and max di-cut problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 67–82. Springer, 2002.

[Mor78] Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, oct 1978.

[NT17] Ashwin Nayak and Dave Touchette. Augmented index and quantum streaming algorithms for dyck(2). In *Proceedings of the 32nd Computational Complexity Conference*, CCC '17, Dagstuhl, DEU, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Rag08] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 245–254, 2008.

[RLS+13] Lee Rhodes, Kevin Lang, Alexander Saydakov, Edo Liberty, and Justin Thaler. Datasketches: A library of stochastic streaming algorithms, 2013.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[Sin23] Noah G Singer. Oblivious algorithms for the Max-kAND problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[SSSV23a] Raghuvansh R. Saxena, Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Improved streaming algorithms for maximum directed cut via smoothed snapshots. In *2023 IEEE 64nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2023.

[SSSV23b] 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 (SODA)*, pages 4083–4103, 2023.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il