Learning-Augmented Streaming Algorithms for Approximating MAX-CUT

Yinhao Dong
School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

Pan Peng
School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

Ali Vakilian
Toyota Technological Institute at Chicago (TTIC), IL, USA

Abstract

We study learning-augmented streaming algorithms for estimating the value of MAX-CUT in a graph. In the classical streaming model, while a $1/2$-approximation for estimating the value of MAX-CUT can be trivially achieved with $O(1)$ words of space, Kapralov and Krachun [STOC'19] showed that this is essentially the best possible: for any $\epsilon > 0$, any (randomized) single-pass streaming algorithm that achieves an approximation ratio of at least $1/2 + \epsilon$ requires $\Omega(n/2^{\text{poly}(1/\epsilon)})$ space.

We show that it is possible to surpass the $1/2$-approximation barrier using just $O(1)$ words of space by leveraging a (machine learned) oracle. Specifically, we consider streaming algorithms that are equipped with an ϵ-accurate oracle that for each vertex in the graph, returns its correct label in $\{-1, +1\}$, corresponding to an optimal MAX-CUT solution in the graph, with some probability $1/2 + \epsilon$, and the incorrect label otherwise.

Within this framework, we present a single-pass algorithm that approximates the value of MAX-CUT to within a factor of $1/2 + \Omega(\epsilon^2)$ with probability at least $2/3$ for insertion-only streams, using only $\text{poly}(1/\epsilon)$ words of space. We also extend our algorithm to fully dynamic streams while maintaining a space complexity of $\text{poly}(1/\epsilon, \log n)$ words.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Learning-Augmented Algorithms, Graph Streaming Algorithms, MAX-CUT

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.44

Related Version *Full Version*: https://arxiv.org/abs/2412.09773

Funding The work was conducted in part while Pan Peng and Ali Vakilian were long-term visitors at the Simons Institute for the Theory of Computing as part of the Sublinear Algorithms program. Yinhao Dong and Pan Peng are supported in part by NSFC grant 62272431.

1 Introduction

Given an undirected, unweighted graph, the MAX-CUT problem seeks to find a partition of the vertices (a cut) that maximizes the number of edges with endpoints on opposite sides of the partition (such edges are said to be "cut" by the partition). MAX-CUT is a well-known NP-hard problem. The best-known approximation algorithm, developed by Goemans and Williamson [22], achieves a 0.878-approximation ratio, which is the best possible under the Unique Games Conjecture [33].

We consider the problem of estimating the value of MAX-CUT in a graph under the streaming model of computation, where MAX-CUT denotes some fixed optimal solution, and the value of a solution is defined as the number of edges in the graph that are cut by the partition. In this model, the graph is presented as a sequence of edge insertions and

© Yinhao Dong, Pan Peng, and Ali Vakilian;
licensed under Creative Commons License CC-BY 4.0

deletions, known as a graph stream. The objective is to analyze the graph's structure using minimal space. The model is called insertion-only if the stream contains only edge insertions; if both insertions and deletions are allowed, it is referred to as the dynamic model.

In the streaming model, a 1/2-approximation for estimating the value of MAX-CUT can be trivially achieved using $O(1)$ words of space, where a word represents the space required to encode the size of the graph. This is done by simply counting the total number m of edges in the graph and outputting $m/2$. However, Kapralov and Krachun [31] showed that this is essentially the best possible: for any $\epsilon > 0$, any (randomized) single-pass streaming algorithm in the insertion-only model that achieves an approximation ratio of at least $1/2+\epsilon$ requires $\Omega(n/2^{\text{poly}(1/\epsilon)})$ space.

We focus on the problem of estimating the value of MAX-CUT in a streaming setting within the framework of *learning-augmented algorithms*. These algorithms utilize predictions from a machine learning model to solve a problem, where the predictions typically include some information about the optimal solution, future events, or yet unread data in the input stream (see, e.g., [24, 27, 13]). Learning-augmented algorithms have gained significant attention recently, partly due to numerous breakthroughs in machine learning. Ideally, such algorithms should be both *robust* and *consistent*: when the predictions are accurate, the algorithm should outperform the best-known classical algorithms, and when the predictions are inaccurate, the algorithm should still provide performance guarantees that are close to or match those of the best-known classical algorithms. Despite the extensive research in the area of learning-augmented algorithms over the past few years[1], our understanding of this framework within the streaming model remains limited.

In this work, we consider the *noisy prediction* model, also referred to as *ϵ-accurate predictions*, where the algorithm has oracle access to a prediction vector $Y \in \{-1,1\}^n$. Each entry Y_v is *independently correct* with probability $\frac{1}{2}+\epsilon$, where $\epsilon \in (0, \frac{1}{2}]$ represents the *bias* of the predictions. Specifically, for each $v \in V$, we have $\Pr[Y_v = x^*_v] = \frac{1}{2}+\epsilon$ and $\Pr[Y_v = -x^*_v] = \frac{1}{2}-\epsilon$, where $x^* \in \{-1,1\}^n$ denotes some fixed but unknown optimal solution for MAX-CUT. This model captures a scenario where an ML algorithm (or oracle) provides predictions for the values of x^* that are unreliable and noisy, being only slightly better than random guesses (i.e., just above the 1/2 probability that a random solution would satisfy). Recently, variants of this prediction model have been used to design improved approximation algorithms for MAX-CUT and constraint satisfaction problems (CSPs) [14, 21].

Specifically, we study the following question:

Given an oracle $\mathcal{O}$ that provides ϵ-accurate predictions Y about the optimal MAX-CUT *of a streaming graph G, can we improve upon the worst-case approximation ratio and space trade-off established by [31], specifically the $(\frac{1}{2}+\epsilon)$ ratio with $\Omega(n)$ space complexity, for estimating the* MAX-CUT *value?*

1.1 Our Results

We provide an affirmative answer by presenting a single-pass streaming algorithm that surpasses the 1/2-approximation barrier for the value of MAX-CUT using only $\text{poly}(1/\epsilon)$ (resp., $\text{poly}(1/\epsilon, \log n)$)words of space in insertion-only (resp., fully dynamic) streams. Formally, we establish the following result:

[1] See `https://algorithms-with-predictions.github.io/` for an up-to-date repository of publications on this topic.

▶ **Theorem 1.** *Let $\epsilon \in (0, \frac{1}{2}]$. Given oracle access to an ϵ-accurate predictor, there exists a single-pass streaming algorithm that provides a $\left(\frac{1}{2} + \Omega(\epsilon^2)\right)$-approximation for estimating the MAX-CUT value of a graph in insertion-only (resp., fully dynamic) streams using $\mathrm{poly}(1/\epsilon)$ (resp., $\mathrm{poly}(1/\epsilon, \log n)$) words of space with probability at least $2/3$.*

By using median trick, we can boost the success probability to $1 - \delta$, where $\delta \in (0, 1)$, with an additional $\log(1/\delta)$ factor in space complexity.

We remark that the "robustness" of our learning-augmented algorithm comes for free, as we can always compute the number of edges in the graph and report half of them as a $\frac{1}{2}$-approximation solution, regardless of the value of ϵ. Furthermore, with predictions, it gains an advantage of $\Omega(\epsilon^2)$ in the approximation ratio. Furthermore, our algorithm does not require the exact value of ϵ; a constant-factor approximation of ϵ is sufficient. This estimation is only needed to determine the sample size.

Our algorithm is based on the observation that when the maximum degree of a graph is relatively small (i.e. smaller than $\mathrm{poly}(\epsilon) \cdot m$), the number of edges with endpoints having different predicted labels already provides a strictly better-than-$\frac{1}{2}$ approximation of the MAX-CUT value. For general graphs, we employ some well-known techniques, such as the CountMin sketch and ℓ_0-sampling, to distinguish between high-degree and low-degree vertices in both insertion-only and dynamic streams. By combining a natural greedy algorithm with an algorithm tailored for the low-degree part, we achieve a non-trivial approximation. The space complexity of the resulting algorithm is primarily determined by the subroutines for identifying the set of high-degree vertices, which can be bounded by $\mathrm{poly}(1/\epsilon, \log n)$.

We note that it is standard to assume that the noisy oracle $\mathcal{O}$ has a compact representation and that its space complexity is not included in the space usage of our algorithm, following the conventions in streaming learning-augmented algorithms [24, 27, 13, 1]. Indeed, as noted in [24], a reliable oracle can often be learned space-efficiently in practice as well. Moreover, we show that in the case of random-order streams, our algorithm only needs to query $\mathcal{O}$ for the labels of a constant number of vertices (see Section B).

Additionally, our algorithm actually works in a weaker model. That is, we can assume that in our streaming framework, the predicted label of each vertex is associated with its edges. Thus, the elements in a stream can be represented as $(e = (u, v), Y_u, Y_v)$, where the predictions $(Y_u)_{u \in V}$ remain consistent throughout the stream. The case of dynamic streams is defined analogously. Notably, in this model, no additional space is required to store the predictors.

1.2 Related Work

Learning-augmented algorithms have been actively researched in online algorithms [42, 35, 4, 6, 7, 25, 41, 5, 36], data structures [43, 20, 46, 39, 44], graph algorithms [18, 12, 8, 37, 16, 10, 40, 23, 17, 14, 21, 11], and sublinear-time algorithms [26, 19, 38, 45]. Our algorithms fit into the category of learning-augmented streaming algorithms. Hsu et al. [24] introduced learning-augmented streaming algorithms for frequency estimation, and Jiang et al. [27] extended this framework to various norm estimation problems in data stream. Recently, Aamand et al. [2, 1] developed learning-augmented frequency estimation algorithms, that improve upon the work of [24]. Additionally, Chen et al. [13] studied single-pass streaming algorithms to estimate the number of triangles and four-cycles in a graph. It is worth mentioning that both our work and previous efforts on learning-augmented streaming algorithms mainly focus on using predictors to improve the corresponding space-accuracy trade-offs. Furthermore, recent studies have explored learning-augmented algorithms for MAX-CUT and constraint satisfaction problems (CSPs) [14, 21] within a variant of our prediction model. In particular, for the Max-CUT problem, Cohen-Addad et al. [14] achieved a $(0.878+\tilde{\Omega}(\epsilon^4))$-approximation using SDP. However, it is important to emphasize that their setting differs significantly from ours: they focus on finding the MAX-CUT in the offline setting, whereas our goal is to estimate its size in the streaming setting. Additionally, Ghoshal et al. [21] developed an algorithm that yields near-optimal solutions when the average degree is larger than a threshold determined only by the bias of the predictions, independent of the instance size. They further extended this result to weighted graphs and the MAX-3LIN problem.

The aforementioned lower bound proven in [31] is, in fact, the culmination of a series of works exploring the trade-offs between space and approximation ratio for the streaming MAX-CUT value problem, including [29, 34, 30]. In our main context, we focus on the problem of estimating the value of MAX-CUT in a streaming setting. Another related problem is that of finding the MAX-CUT itself in a streaming model. Since even outputting the MAX-CUT requires $\Omega(n)$ space, research in this area primarily considers streaming algorithms that use $\tilde{O}(n)$ space, known as the semi-streaming model, where $\tilde{O}(\cdot)$ hides polylogarithmic factors. Beyond the trivial $\frac{1}{2}$-approximation algorithm, which simply partitions the vertices randomly into two parts, there exists a $(1-\epsilon)$-approximation algorithm for finding MAX-CUT. This algorithm works by constructing a $(1-\epsilon)$-cut or spectral sparsifier of the input streaming graph, which can be achieved in $\tilde{O}(n)$ space (see e.g. [3, 32]), and then finding the MAX-CUT of the sparsified graph (which may require exponential time).

2 Preliminaries and Problem Statement

Notations. Throughout the paper we let $G=(V,E)$ be an undirected, unweighted graph with n vertices and m edges. Given a vertex $v \in V$ and disjoint sets $S,T \subseteq V$, $e(v,S)$ denotes the number of edges incident to v whose other endpoint belongs to S; $e(v,S) = |\{(u,v) \mid (u,v) \in E \text{ and } u \in S\}|$. Similarly, we define $e(S,T)$ to denote the number of edges with one endpoint in S and the other endpoint in T; $e(S,T)=\sum_{v\in T} e(v,S)$. A cut (S,T) of G is a bipartition of V into two disjoint subsets S and T. The value of the cut (S,T) is $e(S,T)$.

In the descriptions of our algorithms, for a vertex set $S \subseteq V$, we define S^+ (resp. S^-) as the set of vertices with predicted + (resp. −) signs by the given ϵ-accurate oracle $\mathcal{O}$. Note that S^+ and S^- form a bipartition of S. Throughout the paper, all space complexities are measured in words unless otherwise specified. The space complexity in bits is larger by a factor of $\log m + \log n$.

In our algorithms, we utilize several well-known techniques in the streaming model, which we define below.

CountMin Sketch [15]. Given a stream of m integers from $[n]$, let f_i denote number of occurrences of i in the stream, for any $i \in [n]$. In CountMin sketch, there are k uniform and independent random hash functions $h_1, h_2, \dots, h_k : [n] \to [w]$ and an array C of size $k \times w$. The algorithm maintains C, such that $C[\ell, s] = \sum_{j:h_\ell(j)=s} f_j$ at the end of the stream. Upon each query of an arbitrary $i \in [n]$, the algorithm returns $\tilde{f}_i = \min_{\ell\in[k]} C[\ell, h_\ell(i)]$.

▶ **Theorem 2** (CountMin Sketch [15]). *For any $i \in [n]$ and any $\ell \in [k]$, we have $\tilde{f}_i \geq f_i$ and $\mathbb{E}[C[\ell, h_\ell(i)]] \leq f_i + \frac{m}{w}$. The space complexity is $O(kw)$ words. Let $\epsilon, \delta \in (0,1)$. If we set $k = \lceil \frac{e}{\epsilon} \rceil$ and $w = \lceil \ln \frac{1}{\delta} \rceil$, then for any $i \in [n]$, we have $\tilde{f}_i \geq f_i$, and with probability at least $1-\delta$, $\tilde{f}_i \leq f_i + \epsilon m$. The corresponding space complexity is $O(\frac{1}{\epsilon} \ln \frac{1}{\delta})$ words.*

Reservoir Sampling [47]. The algorithm samples k elements from a stream, such that at any time $m \geq k$, the sample consists of a uniformly random subset of size k of the elements seen so far. The space complexity of the algorithm is $O(k(\log n + \log m))$ bits.

▶ **Definition 3** (ℓ_0-sampler [28]). *Let $x \in \mathbb{R}^n$ be a non-zero vector and $\delta \in (0, 1)$. An ℓ_0-sampler for x returns FAIL with probability at most δ and otherwise returns some index i such that $x_i \neq 0$ with probability $\frac{1}{|\text{supp}(x)|}$ where $\text{supp}(x) = \{i \mid x_i \neq 0\}$ is the support of x.*

The following theorem states that ℓ_0-samplers can be maintained using a single pass in dynamic streams.

▶ **Theorem 4** (Theorem 2 in [28]). *Let $\delta \in (0, 1)$. There exists a single-pass streaming algorithm for maintaining an ℓ_0-sampler for a non-zero vector $x \in \mathbb{R}^n$ (with failure pribability δ) in the dynamic model using $O(\log^2 n \log \delta^{-1})$ bits of space.*

Problem Statement. In this work, we consider the problem of estimating the value of MAX-CUT of a graph G in the learning-augmented streaming model. Formally, given an undirected, unweighted graph $G = (V, E)$, which is represented as a sequence of edges, i.e., $\sigma := \langle e_1, e_2, \ldots \rangle$ for $e_i \in E$, our goal is to scan the sequence in one pass and output an estimate of the MAX-CUT value, while minimizing space usage. When the sequence contains only edge insertions, it is referred to as an *insertion-only* stream. When the sequence contains both edge insertions and deletions, it is referred to as a *dynamic* stream. Note that in dynamic streams, the sequence of the stream is often represented as $\sigma := \langle (e_1, \Delta_1), (e_2, \Delta_2), \ldots \rangle$, where for each i, $e_i \in E$ and $\Delta_i = 1$ (resp. -1) denotes edge insertion (resp. deletion).

Furthermore, we assume that the algorithms are equipped with an oracle $\mathcal{O}$, which provides ϵ-accurate predictions $Y \in \{-1, 1\}^n$ where $\epsilon \in (0, \frac{1}{2}]$. This information is provided via an external oracle, and for the purpose of our algorithm, we assume that the edges in the stream are annotated with the predictions of their endpoints.

Specifically, for each vertex $v \in V$, $\Pr[Y_v = x_v^*] = \frac{1}{2} + \epsilon$ and $\Pr[Y_v = -x_v^*] = \frac{1}{2} - \epsilon$, where $x^* \in \{-1, 1\}^n$ is some fixed but unknown optimal solution. Following previous works on MAX-CUT with ϵ-accurate predictions [14, 21], we also assume that the $(Y_v)_{v \in V}$ are independent.

Finally, since we assume that accuracy parameter of predictions, ϵ, is known to the algorithm up to a constant factor in advance, and the space complexity of all algorithms in the paper is at most $\text{poly}(\log n, 1/\epsilon, 1/\delta)$, we will assume throughout that $m = \Omega(\epsilon^{-11}\delta^{-7})$. Otherwise, when $m = O(\epsilon^{-11}\delta^{-7})$, we can simply store all edges in $\text{poly}(1/\epsilon, 1/\delta)$ space and compute the MAX-CUT exactly.

3 The Simple Case of Low-Degree Graphs

In this section, we show that when all vertices have "low-degree" then following the ϵ-accurate predictions, we can beat the $\frac{1}{2}$-approximation barrier for streaming MAX-CUT. Informally, once an edge (u, v) arrives in the stream, we consult the noisy oracle $\mathcal{O}$ and if their endpoints are predicted to be in different parts by the noisy oracle, i.e., $\mathcal{O}(u) \neq \mathcal{O}(v)$, we increase the size of the cut by one. This can be simply implemented using $O(\log n)$ bits of space. Refer to Algorithm 1 for a formal description.

Next, we analyze the space complexity and approximation guarantee of Algorithm 1.

Algorithm 1 A learning-augmented streaming algorithm for estimating MAX-CUT value in low-degree graphs.

Input: Graph G with $\Delta < \frac{\epsilon^2 \delta m}{4} - \frac{1}{4\epsilon^2}$ as a stream of edges, an ϵ-accurate oracle $\mathcal{O} \to \{-1, 1\}$
Output: Estimate of the MAX-CUT value of G

1: **initialize** $X \leftarrow 0$.
2: **for each** edge (u, v) in the stream **do**
3: $\quad Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
4: $\quad$ **if** $Y_u \neq Y_v$ **then** $X \leftarrow X + 1$
5: **return** X

▶ **Theorem 5.** *Let $\epsilon \in (0, \frac{1}{2}]$ and $\delta \in (0, 1)$. Given an ϵ-accurate oracle $\mathcal{O}$ for the MAX-CUT of G, if the maximum degree Δ in G satisfies $\Delta < \frac{\epsilon^2 \delta m}{4} - \frac{1}{4\epsilon^2}$, then with probability at least $1 - \delta$, Algorithm 1 outputs a $(\frac{1}{2} + \epsilon^2)$-approximation of the MAX-CUT value of G. The algorithm uses $O(1)$ words of space.*

Proof. We first analyze the space complexity. In Algorithm 1, we maintain a counter X for the edges whose endpoints have different signs, which takes $O(\log n)$ bits of space. Therefore, the space complexity of the algorithm is $O(1)$ words.

Next, we show that Algorithm 1 with high probability outputs a $(\frac{1}{2} + \epsilon^2)$-approximation of the MAX-CUT when all vertices have low degrees.

Since (V^+, V^-) is a feasible cut of G, where V^+ (resp. V^-) is the set of vertices with predicted $+$ (resp. $-$) signs by the given ϵ-accurate oracle $\mathcal{O}$, we have $X \leq \mathrm{OPT}$, where OPT denotes the (optimal) MAX-CUT value of G. For each edge $(u, v) \in E$, define an indicator random variable X_{uv} such that $X_{uv} = 1$ if $Y_u \neq Y_v$, and is zero otherwise.

Let x^* be the assignment vector corresponding to the MAX-CUT in G. Specifically, if $(S, V \setminus S)$ is the MAX-CUT of G, then $x_v^* = 1$ if $v \in S$, and -1 otherwise. Consider the following two cases:

(I.1) If $x_u^* \neq x_v^*$, then $\Pr[Y_u \neq Y_v] = \Pr[Y_u = x_u^* \wedge Y_v = x_v^*] + \Pr[Y_u \neq x_u^* \wedge Y_v \neq x_v^*] = \frac{1}{2} + 2\epsilon^2$.

(I.2) If $x_u^* = x_v^*$, then $\Pr[Y_u \neq Y_v] = \Pr[Y_u = x_u^* \wedge Y_v \neq x_v^*] + \Pr[Y_u \neq x_u^* \wedge Y_v = x_v^*] = \frac{1}{2} - 2\epsilon^2$.

Note that $X = \sum_{(u,v) \in E} X_{uv}$ and $\mathbb{E}[X_{uv}] = \Pr[Y_u \neq Y_v]$. Then, by the linearity of expectation,

$$\begin{aligned} \mathbb{E}[X] = \sum_{\substack{(u,v) \in E \\ x_u^* \neq x_v^*}} \mathbb{E}[X_{uv}] + \sum_{\substack{(u,v) \in E \\ x_u^* = x_v^*}} \mathbb{E}[X_{uv}] &= \mathrm{OPT} \cdot (\frac{1}{2} + 2\epsilon^2) + (m - \mathrm{OPT}) \cdot (\frac{1}{2} - 2\epsilon^2) \\ &= (\frac{1}{2} - 2\epsilon^2) m + 4\epsilon^2 \cdot \mathrm{OPT} \\ &\geq (\frac{1}{2} + 2\epsilon^2) \cdot \mathrm{OPT}, \end{aligned}$$

where the last inequality holds since $\frac{1}{2} - 2\epsilon^2 \geq 0$ for $\epsilon \in (0, \frac{1}{2}]$ and $m \geq \mathrm{OPT}$.

Next, we bound $\mathrm{Var}[X]$. Note that $\mathrm{Var}[X_{uv}] = \mathbb{E}[X_{uv}] - \mathbb{E}[X_{uv}]^2$ since $(X_{uv})_{(u,v) \in E}$ are indicator random variables. By (I.1) and (I.2), $\mathrm{Var}[X_{uv}] = \frac{1}{4} - 4\epsilon^4$.

(II.1) If $x_u^* \neq x_v^*$, then $\mathrm{Var}[X_{uv}] = \left(\frac{1}{2} + 2\epsilon^2\right) - \left(\frac{1}{2} + 2\epsilon^2\right)^2 = \frac{1}{4} - 4\epsilon^4$.

(II.2) If $x_u^* = x_v^*$, then $\mathrm{Var}[X_{uv}] = \left(\frac{1}{2} - 2\epsilon^2\right) - \left(\frac{1}{2} - 2\epsilon^2\right)^2 = \frac{1}{4} - 4\epsilon^4$.

For any pair of distinct edges $(u,v),(u,w) \in E$, we compute $\mathbb{E}[X_{uv}X_{uw}] = \Pr[X_{uv} = X_{uw} = 1] = \Pr[Y_v = Y_w = -Y_u]$ and $\mathrm{Cov}[X_{uv}, X_{uw}] = \mathbb{E}[X_{uv}X_{uw}] - \mathbb{E}[X_{uv}] \cdot \mathbb{E}[X_{uw}]$.

(III.1) If $x_u^* = x_v^*$ and $x_u^* = x_w^*$, then

$$\begin{aligned}\mathbb{E}[X_{uv}X_{uw}] &= \Pr[Y_u = x_u^* \wedge Y_v \neq x_v^* \wedge Y_w \neq x_w^*] + \Pr[Y_u \neq x_u^* \wedge Y_v = x_v^* \wedge Y_w = x_w^*] \\ &= \left(\frac{1}{2}+\epsilon\right)\left(\frac{1}{2}-\epsilon\right)^2 + \left(\frac{1}{2}-\epsilon\right)\left(\frac{1}{2}+\epsilon\right)^2 = \frac{1}{4} - \epsilon^2.\end{aligned}$$

Therefore, $\mathrm{Cov}[X_{uv}, X_{uw}] = \left(\frac{1}{4} - \epsilon^2\right) - \left(\frac{1}{2} - 2\epsilon^2\right)^2 = \epsilon^2 - 4\epsilon^4$.

(III.2) If $x_u^* = x_v^*$ and $x_u^* \neq x_w^*$, then

$$\begin{aligned}\mathbb{E}[X_{uv}X_{uw}] &= \Pr[Y_u = x_u^* \wedge Y_v \neq x_v^* \wedge Y_w = x_w^*] + \Pr[Y_u \neq x_u^* \wedge Y_v = x_v^* \wedge Y_w \neq x_w^*] \\ &= \left(\frac{1}{2}+\epsilon\right)^2\left(\frac{1}{2}-\epsilon\right) + \left(\frac{1}{2}-\epsilon\right)^2\left(\frac{1}{2}+\epsilon\right) = \frac{1}{4} - \epsilon^2.\end{aligned}$$

Therefore, $\mathrm{Cov}[X_{uv}, X_{uw}] = \left(\frac{1}{4} - \epsilon^2\right) - \left(\frac{1}{2} - 2\epsilon^2\right)\left(\frac{1}{2} + 2\epsilon^2\right) = 4\epsilon^4 - \epsilon^2$.

(III.3) If $x_u^* \neq x_v^*$ and $x_u^* = x_w^*$, then

$$\begin{aligned}\mathbb{E}[X_{uv}X_{uw}] &= \Pr[Y_u = x_u^* \wedge Y_v = x_v^* \wedge Y_w \neq x_w^*] + \Pr[Y_u \neq x_u^* \wedge Y_v \neq x_v^* \wedge Y_w = x_w^*] \\ &= \left(\frac{1}{2}+\epsilon\right)^2\left(\frac{1}{2}-\epsilon\right) + \left(\frac{1}{2}-\epsilon\right)^2\left(\frac{1}{2}+\epsilon\right) = \frac{1}{4} - \epsilon^2.\end{aligned}$$

Therefore, $\mathrm{Cov}[X_{uv}, X_{uw}] = \left(\frac{1}{4} - \epsilon^2\right) - \left(\frac{1}{2} + 2\epsilon^2\right)\left(\frac{1}{2} - 2\epsilon^2\right) = 4\epsilon^4 - \epsilon^2$

(III.4) If $x_u^* \neq x_v^*$ and $x_u^* \neq x_w^*$, then

$$\begin{aligned}\mathbb{E}[X_{uv}X_{uw}] &= \Pr[Y_u = x_u^* \wedge Y_v = x_v^* \wedge Y_w = x_w^*] + \Pr[Y_u \neq x_u^* \wedge Y_v \neq x_v^* \wedge Y_w \neq x_w^*] \\ &= \left(\frac{1}{2}+\epsilon\right)^3 + \left(\frac{1}{2}-\epsilon\right)^3 = \frac{1}{4} + 3\epsilon^2.\end{aligned}$$

Therefore, $\mathrm{Cov}[X_{uv}, X_{uw}] = \left(\frac{1}{4} + 3\epsilon^2\right) - \left(\frac{1}{2} + 2\epsilon^2\right)^2 = \epsilon^2 - 4\epsilon^4$.

Hence, $\mathrm{Cov}[X_{uv}, X_{uw}] \leq \epsilon^2 - 4\epsilon^4$. So,

$$\begin{aligned}\mathrm{Var}[X] &= \sum_{(u,v)\in E} \mathrm{Var}[X_{uv}] + \sum_{\substack{(u,v),(u,w)\in E \\ v\neq w}} \mathrm{Cov}[X_{uv}, X_{uw}] + \sum_{\substack{(u,v),(w,z)\in E \\ u,v,w,z \text{ all distinct}}} \mathrm{Cov}[X_{uv}, X_{wz}] \\ &\leq m \cdot \left(\frac{1}{4} - 4\epsilon^4\right) + m \cdot \Delta \cdot \left(\epsilon^2 - 4\epsilon^4\right) + 0 \leq \left(\frac{1}{4} + \Delta\epsilon^2\right) m.\end{aligned}$$

Then, by applying Chebyshev's inequality,

$$\begin{aligned}\Pr\left[X \leq \left(\frac{1}{2}+\epsilon^2\right)\mathrm{OPT}\right] &\leq \Pr\left[|X - \mathbb{E}[X]| \geq \epsilon^2 \cdot \mathrm{OPT}\right] \\ &\leq \frac{\mathrm{Var}[X]}{\epsilon^4 \cdot \mathrm{OPT}^2} \\ &\leq \frac{\left(\frac{1}{4} + \Delta\epsilon^2\right) m}{\epsilon^4 \cdot \mathrm{OPT}^2} && \triangleright\ \mathrm{Var}[X] \leq (\frac{1}{4} + \Delta\epsilon^2)m \\ &\leq \frac{4 \cdot (\frac{1}{4} + \Delta\epsilon^2)}{\epsilon^4 m} && \triangleright\ \mathrm{OPT} \geq \frac{m}{2} \\ &< \delta. && \triangleright\ \Delta < \frac{\epsilon^2 \delta m}{4} - \frac{1}{4\epsilon^2}\end{aligned}$$

So, with probability at least $1-\delta$, $X \geq (\frac{1}{2} + \epsilon^2) \cdot \mathrm{OPT}$. ◀

4 Our Algorithm for General Graphs

In this section, we present our $(\frac{1}{2}+\Omega(\epsilon^2))$-approximation for streaming MAX-CUT in $O(\text{poly}(1/\epsilon, 1/\delta))$ space using ϵ-accurate predictions. For better presenting our algorithmic ideas, we first consider the problem in the *random order* streams in Section 4.1. Next, in Section 4.2, we show how to extend our algorithm to arbitrary order streams using more advanced sketching techniques. Finally, in Section 4.3, we show that our algorithm also work in dynamic streams where both edge insertions and deletions are allowed.

Offline Implementation. We first describe our algorithm in the offline setting. Recall that our algorithm from Section 3 is effective when the maximum degree in the graph is smaller than a pre-specified threshold $\phi = \Theta(\epsilon^2\delta m)$, where δ is the target failing probability of the algorithm. Let H and L respectively denote the high-degree (i.e., vertices with degree at least ϕ) and low-degree (i.e., vertices with degree less than ϕ) vertices in the input graph $G = (V, E)$. Now, by the guarantee of Theorem 5, suppose we have a $(\frac{1}{2}+\epsilon^2)$-approximation of MAX-CUT on the induced subgraph $G[L]$ denoted by (L^+, L^-). This cut is exactly the cut suggested by the given ϵ-accurate oracle $\mathcal{O}$. Next, we extend this cut using the vertices in H in a greedy manner. We pick vertices in H one by one in an arbitrary order and each time add them to either L^+ or L^- sections so that the size of cut maximized. We denote the resulting cut by $(C, V\setminus C)$. Another cut we consider is simply (H, L). We show at least one these two cuts is a $(\frac{1}{2}+\Omega(\epsilon^2))$-approximation for MAX-CUT on G with high probability.

Throughout the paper, we define high-degree vertices as those with degree $\geq \frac{\epsilon^2 m}{c}$, where $c = \frac{80}{\delta}$. Vertices with degrees below this threshold are considered low-degree.

▶ **Theorem 6.** *Let $\epsilon \in (0, \frac{1}{2}]$ and $\delta \in (0,1)$. The best of $(C, V\setminus C)$ and (H, L) cuts is a $(\frac{1}{2}+\frac{\epsilon^2}{16})$-approximation for MAX-CUT on G with probability at least $1-\delta$.*

Without loss of generality, we assume that $e(H,L) < (\frac{1}{2}+\epsilon^2)\cdot \text{OPT}$, otherwise we are done. Then it suffices to show that $e(C, V\setminus C) \geq (\frac{1}{2}+\frac{\epsilon^2}{16})\cdot\text{OPT}$. Recall that the cut $(C, V\setminus C)$ is obtained by extending the cut (L^+, L^-) of $G[L]$ suggested by the given ϵ-accurate oracle using the vertices in H in a greedy manner. We first show that Algorithm 1 works for the induced subgraph $G[L]$.

▶ **Lemma 7.** *Let $\epsilon \in (0, \frac{1}{2}]$ and $\delta \in (0,1)$. Suppose that $e(H,L) < (\frac{1}{2}+\epsilon^2)\cdot\text{OPT}$, $c = \frac{80}{\delta}$ and $m > \frac{c^3\delta}{\epsilon^4} + \frac{1}{4\epsilon^4}$. Then there exists a $(\frac{1}{2}+\epsilon^2)$-approximation of the MAX-CUT value of $G[L]$ with probability at least $1-\delta$.*

Proof. Note that $m = m_L + m_H + e(H,L)$, where m_L (resp. m_H) is the number of edges in $G[L]$ (resp. $G[H]$). Then we have $m_L = m - m_H - e(H,L) > m - \frac{4c^2}{\epsilon^4} - (\frac{1}{2}+\epsilon^2)m$, since $|H| \leq \frac{2m}{\epsilon^2 m/c} = \frac{2c}{\epsilon^2}$ and $e(H,L) < (\frac{1}{2}+\epsilon^2)\cdot\text{OPT} \leq (\frac{1}{2}+\epsilon^2)m$.

Since the high-degree threshold is $\frac{\epsilon^2 m}{c}$, we have $\Delta_L < \frac{\epsilon^2 m}{c}$, where Δ_L is the maximum degree of $G[L]$. It is easy to check that $\frac{\epsilon^2 m}{c} < \frac{\epsilon^2\delta}{4}(m - \frac{4c^2}{\epsilon^4} - (\frac{1}{2}+\epsilon^2)m) - \frac{1}{4\epsilon^2}$ by substituting in the conditions for c and m. It follows that $\Delta_L < \frac{\epsilon^2\delta m_L}{4} - \frac{1}{4\epsilon^2}$.

Therefore, by the guarantee of Theorem 5, with probability at least $1-\delta$, there exists a $(\frac{1}{2}+\epsilon^2)$-approximation of MAX-CUT on $G[L]$ denoted by (L^+, L^-), i.e., $e(L^+, L^-) \geq (\frac{1}{2}+\epsilon^2)\cdot\text{OPT}_L$, where OPT_L is the size of the MAX-CUT value of $G[L]$. ◀

▶ **Lemma 8.** *Let $\epsilon \in (0, \frac{1}{2}]$. Suppose that $e(H,L) = \alpha\cdot\text{OPT}$ (where $\alpha < \frac{1}{2}+\epsilon^2$) and $m > \frac{8c^2}{\epsilon^8}$. We have $\text{OPT}_L > (1-\alpha-\epsilon^4)\cdot\text{OPT}$, where OPT_L is the size of the MAX-CUT value of $G[L]$.*

Proof. Note that $\text{OPT} \le \text{OPT}_L + \text{OPT}_H + e(H,L)$, where OPT_L (resp. OPT_H) is the size of the MAX-CUT of $G[L]$ (resp. $G[H]$). It follows that

$$\begin{aligned}
\text{OPT}_L &\ge \text{OPT} - \text{OPT}_H - e(H,L) \\
&\ge \text{OPT} - \frac{4c^2}{\epsilon^4} - e(H,L) && \triangleright\ \text{OPT}_H \le m_H \le |H|^2 \le \frac{4c^2}{\epsilon^4} \\
&= \text{OPT} - \frac{4c^2}{\epsilon^4} - \alpha \cdot \text{OPT} && \triangleright\ e(H,L) = \alpha \cdot \text{OPT} \\
&> \text{OPT} - \epsilon^4 \cdot \text{OPT} - \alpha \cdot \text{OPT} && \triangleright\ \frac{4c^2}{\epsilon^4} < \epsilon^4 \cdot \frac{m}{2} \le \epsilon^4 \cdot \text{OPT} \\
&= (1 - \alpha - \epsilon^4) \cdot \text{OPT}.
\end{aligned}$$

◀

Proof of Theorem 6. Since the algorithm returns the best of two cuts, if the value of the cut (H,L) is at least $(\frac{1}{2} + \epsilon^2) \cdot \text{OPT}$ (i.e., $e(H,L) \ge (\frac{1}{2} + \epsilon^2) \cdot \text{OPT}$), then we are done. Hence, without loss of generality, we can assume that the size of the cut (H,L) is strictly less than $(\frac{1}{2} + \epsilon^2) \cdot \text{OPT}$. Based on Lemma 7 and Lemma 8, we have

$$\begin{aligned}
e(C, V \setminus C) &\ge e(L^+, L^-) + \sum_{v \in H} \max\{e(v, L^+), e(v, L^-)\} \\
&\ge \left(\frac{1}{2} + \epsilon^2\right) \text{OPT}_L + \sum_{v \in H} \max\{e(v, L^+), e(v, L^-)\} && \triangleright\ e(L^+, L^-) \ge (\frac{1}{2} + \epsilon^2)\text{OPT}_L \\
&\ge \left(\frac{1}{2} + \epsilon^2\right) \text{OPT}_L + \frac{1}{2} \cdot e(H,L) && \triangleright\ \max\{e(v, L^+), e(v, L^-)\} \ge \frac{e(v,L)}{2} \\
&> \left(\frac{1}{2} + \epsilon^2\right) (1 - \alpha - \epsilon^4)\text{OPT} + \frac{\alpha}{2} \cdot \text{OPT} && \triangleright\ \text{OPT}_L > (1 - \alpha - \epsilon^4)\text{OPT} \\
&= \left(\frac{1}{2} + (1-\alpha)\epsilon^2 - \frac{\epsilon^4}{2} - \epsilon^6\right) \text{OPT} \\
&> \left(\frac{1}{2} + \left(\frac{1}{2} - \epsilon^2\right)\epsilon^2 - \frac{\epsilon^4}{2} - \epsilon^6\right) \text{OPT} && \triangleright\ \alpha < \frac{1}{2} + \epsilon^2 \\
&\ge \left(\frac{1}{2} + \frac{\epsilon^2}{16}\right) \text{OPT}, && \triangleright\ \left(\frac{1}{2} - \epsilon^2\right)\epsilon^2 - \frac{\epsilon^4}{2} - \epsilon^6 \ge \frac{\epsilon^2}{16}
\end{aligned}$$

with probability at least $1 - \delta$. ◀

4.1 Warm-up: Random Order Streams

Overview of the Algorithm. Suppose that the edges of $G = (V, E)$ arrive one by one in a random order stream. At a high level, we would like to run our algorithm from Section 3 for low-degree vertices L (i.e., the induced subgraph on L), and then, using a greedy approach, add the high-degree vertices H to the constructed cut. However, the set H and L are not known a priori in the stream, and it is not clear how to store the required information to run the described algorithm, space efficiently.

To detect high-degree vertices and collect sufficient information to run the greedy approach on them at the end of the stream, we rely on the random order of the stream. We store a small number of edges from the beginning of the stream and then gather degree information for those vertices that are candidates for being high-degree. More precisely, we store the first $\text{poly}(1/\epsilon, 1/\delta)$ edges in the (random order) stream and use them to identify a set $\tilde{H}$ of size $\text{poly}(1/\epsilon, 1/\delta)$ that contains all high-degree vertices H with probability $1 - \delta$. We then store all edges between vertices in $\tilde{H}$ throughout the stream, which requires $\text{poly}(1/\epsilon, 1/\delta)$ words of space. Additionally, for every vertex $v \in \tilde{H}$, we maintain the number of edges between v

and $\tilde{L}^+ := V^+ \setminus \tilde{H}$, as well as between v and $\tilde{L}^- := V^- \setminus \tilde{H}$. It is straightforward to check that these counters require poly$(1/\epsilon, 1/\delta)$ words of space in total. We then apply Algorithm 1 to the graph $G[V \setminus \tilde{H}]$ to approximate the MAX-CUT value of $G[V \setminus \tilde{H}]$.

Finally, at the end of stream, since we can compute the degree of all vertices in $\tilde{H}$ exactly, we can determine the set of high degree vertices $H \subseteq \tilde{H}$. First, for the remaining vertices $\tilde{S} := \tilde{H} \setminus H$, we (hypothetically) feed them to our algorithm for the low-degree vertices. Note that we have all edges between any pair of vertices in $\tilde{S}$, as well as all the number of their incident edges to $\tilde{L}^+$ and $\tilde{L}^-$. Therefore, we can compute the size of (L^+, L^-) cut exactly. Now it only remains to run the greedy algorithm for the high-degree vertices H. Similarly, since we have computed the number of incident edges of high-degree vertices to $\tilde{L}^+, \tilde{L}^-$, and we have stored all edges with both endpoints in $\tilde{H}$, we can perform the greedy extension of $(\tilde{L}^+ \cup \tilde{S}^+, \tilde{L}^- \cup \tilde{S}^-)$ using H. Moreover, using the same set of degree information, we can compute the size of $(H, V \setminus H)$ as well. Then, we can return the best of these two cuts as our estimate of MAX-CUT in G.

Next, we prove the approximation guarantee and the space complexity of the algorithm. The algorithm is formally given in Algorithm 2.

▶ **Theorem 9.** *Let $\epsilon \in (0, \frac{1}{2}]$ and $\delta \in (0, 1)$. Given an ϵ-accurate oracle $\mathcal{O}$ for the MAX-CUT of G, there exists a single-pass $(\frac{1}{2} + \frac{\epsilon^2}{16})$-approximation algorithm for estimating the MAX-CUT value of G in the insertion-only random-order streams. The algorithm uses $O(\frac{1}{\delta^6 \epsilon^8})$ words of space. The approximation holds with probability at least $1 - \delta$.*

▶ **Lemma 10.** *Let $\delta \in (0, 1)$. Then $\tilde{H}$ contains all high-degree vertices in G with probability at least $1 - \frac{\delta}{2}$.*

Proof. Suppose that v is a high-degree vertex in G, i.e., $\deg(v) \geq \frac{\epsilon^2 m}{c}$. For each edge e_i incident to v, define an indicator random variable

$$X_i = \begin{cases} 1, & \text{if } e_i \text{ is in the first } \frac{\beta}{\delta^3 \epsilon^4} \text{ edges of the random order stream} \\ 0, & \text{otherwise} \end{cases}$$

So, $\mathbb{E}[X_i] = \Pr[X_i = 1] = \frac{\beta}{\delta^3 \epsilon^4 m}$. We define $X := \sum_{i \in [\deg(v)]} X_i$ to denote the number of edges incident to v that appear in the first $\frac{\beta}{\delta^3 \epsilon^4}$ edges of the random order stream. Then, $\mathbb{E}[X] = \sum_{i \in [\deg(v)]} \mathbb{E}[X_i] = \frac{\beta \deg(v)}{\delta^3 \epsilon^4 m}$.

Next, we bound $\mathrm{Var}[X]$. For any $i \in [\deg(v)]$, $\mathrm{Var}[X_i] = \mathbb{E}[X_i^2] - (\mathbb{E}[X_i])^2 = \mathbb{E}[X_i] - (\mathbb{E}[X_i])^2 = \frac{\beta}{\delta^3 \epsilon^4 m} - (\frac{\beta}{\delta^3 \epsilon^4 m})^2 = \frac{\beta}{\delta^3 \epsilon^4 m} - \frac{\beta^2}{\delta^6 \epsilon^8 m^2}$. Since $(X_i)_{i \in [\deg(v)]}$ are negatively correlated, $\mathrm{Var}[X] \leq \sum_{i \in [\deg(v)]} \mathrm{Var}[X_i] = \deg(v) \cdot \left(\frac{\beta}{\delta^3 \epsilon^4 m} - \frac{\beta^2}{\delta^6 \epsilon^8 m^2}\right) \leq \frac{\beta \deg(v)}{\delta^3 \epsilon^4 m}$. Therefore, by Chebyshev's inequality,

$$\begin{aligned} \Pr[v \notin \tilde{H}] = \Pr[X \leq 0] &= \Pr\left[X \leq \mathbb{E}[X] - \frac{\beta \deg(v)}{\delta^3 \epsilon^4 m}\right] \\ &\leq \frac{\mathrm{Var}[X] \cdot \delta^6 \epsilon^8 m^2}{\beta^2 \deg^2(v)} \\ &\leq \frac{\delta^3 \epsilon^4 m}{\beta \cdot \deg(v)} && \triangleright\ \mathrm{Var}[X] \leq \frac{\beta \deg(v)}{\delta^3 \epsilon^4 m} \\ &\leq \frac{\delta^3 \epsilon^2 c}{\beta}. && \triangleright\ \deg(v) \geq \frac{\epsilon^2 m}{c} \end{aligned}$$

By our definition of high-degree vertices, there are at most $\frac{2m}{\epsilon^2 m / c} = \frac{2c}{\epsilon^2}$ high-degree vertices in G. By union bound, with probability at least $1 - \frac{2c}{\epsilon^2} \cdot \frac{\delta^3 \epsilon^2 c}{\beta} = 1 - \frac{2c^2 \delta^3}{\beta} \geq 1 - \frac{\delta}{2}$ (since $c = \frac{80}{\delta}$ and β is sufficiently large), $\tilde{H}$ contains all high-degree vertices in G. ◀

Algorithm 2 Estimating the MAX-CUT value in (insertion-only) random-order streams.

Input: Graph G as a random-order stream of edges, an ϵ-accurate oracle $\mathcal{O} \to \{-1, 1\}$, a high-degree threshold $\theta = \frac{\epsilon^2 m}{c}$ (where $c = \frac{80}{\delta}$).
Output: The estimate of the MAX-CUT value of G.

▷ **Preprocessing phase**
1: **initialize** $F \leftarrow \emptyset$, $\tilde{H} \leftarrow \emptyset$, $e(L^+, L^-) \leftarrow 0$.
▷ **Streaming phase**
2: **for each** edge (u, v) in the first $\frac{\beta}{\delta^3 \epsilon^4}$ edges of the stream (where β is a sufficiently large universal constant) **do**
3: $\quad F \leftarrow F \cup \{(u, v)\}$, $\tilde{H} \leftarrow \tilde{H} \cup \{u, v\}$
4: **for each** vertex $v \in \tilde{H}$ **do**
5: $\quad$ **initialize** $e(v, V \setminus \tilde{H}) \leftarrow 0$, $e(v, L^+) \leftarrow 0$, $e(v, L^-) \leftarrow 0$.
6: **for each** remaining edge (u, v) in the stream **do**
7: $\quad Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
8: $\quad$ **if** $u \in \tilde{H}$ and $v \in \tilde{H}$ **then** $F \leftarrow F \cup \{(u, v)\}$
9: $\quad$ **else**
10: $\quad\quad$ **if** $u \in \tilde{H}$ and $v \in V \setminus \tilde{H}$ **then** $e(u, V \setminus \tilde{H}) \leftarrow e(u, V \setminus \tilde{H}) + 1$
11: $\quad\quad$ **if** $v \in \tilde{H}$ and $u \in V \setminus \tilde{H}$ **then** $e(v, V \setminus \tilde{H}) \leftarrow e(v, V \setminus \tilde{H}) + 1$
12: $\quad$ **if** $u \in \tilde{H}$ and $v \in V \setminus \tilde{H}$ **then**
13: $\quad\quad$ **if** $Y_v = 1$ **then** $e(u, L^+) \leftarrow e(u, L^+) + 1$ **else** $e(u, L^-) \leftarrow e(u, L^-) + 1$
14: $\quad$ **if** $u \in V \setminus \tilde{H}$ and $v \in V \setminus \tilde{H}$ and $Y_u \neq Y_v$ **then**
15: $\quad\quad e(L^+, L^-) \leftarrow e(L^+, L^-) + 1$ (i.e., apply Algorithm 1 on $G[V \setminus \tilde{H}]$)
▷ **Postprocessing phase**
16: $H \leftarrow \{v \in \tilde{H} : |\{e \in F : v \in e\}| + e(v, V \setminus \tilde{H}) \geq \theta\}$, $L \leftarrow V \setminus H$
17: **for each** edge $(u, v) \in F$ **do**
18: $\quad Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
19: $\quad$ **if** $u \in H$ and $v \in L$ **then**
20: $\quad\quad$ **if** $Y_v = 1$ **then** $e(u, L^+) \leftarrow e(u, L^+) + 1$ **else** $e(u, L^-) \leftarrow e(u, L^-) + 1$
21: $\quad$ **if** $u \in \tilde{H} \setminus H$ and $v \in \tilde{H} \setminus H$ and $Y_u \neq Y_v$ **then** $e(L^+, L^-) \leftarrow e(L^+, L^-) + 1$
22: **for each** vertex $v \in \tilde{H} \setminus H$ **do**
23: $\quad Y_v = \mathcal{O}(v)$
24: $\quad$ **if** $Y_v = 1$ **then** $e(L^+, L^-) \leftarrow e(L^+, L^-) + e(v, L^-)$ **else** $e(L^+, L^-) \leftarrow e(L^+, L^-) + e(v, L^+)$
25: $\text{ALG}_1 \leftarrow e(L^+, L^-) + \sum_{v \in H} \max\{e(v, L^-), e(v, L^+)\}$
26: $\text{ALG}_2 \leftarrow \sum_{v \in H} (e(v, L^-) + e(v, L^+))$
27: **return** $\max\{\text{ALG}_1, \text{ALG}_2\}$

Proof of Theorem 9. In Algorithm 2, we store the first $\frac{\beta}{\delta^3 \epsilon^4}$ edges of the stream (Line 3) and the remaining edges with both endpoints in $\tilde{H}$ (Line 8). We also maintain several counters for vertices in $\tilde{H}$. Therefore, the total space complexity of Algorithm 2 is $O(\frac{1}{\delta^6 \epsilon^8})$ words.

Next, we show the approximation guarantee of Algorithm 2. Note that at the end of the stream we can identify H exactly, using the information stored during the streaming phase. In Algorithm 2, we apply Algorithm 1 to the subgraph $G[V \setminus \tilde{H}]$ during the streaming phase (Line 15). Hypothetically, during the postprocessing phase, we apply Algorithm 1 to the subgraph $G[\tilde{H} \setminus H]$ (Line 21) and to edges with one endpoint in $\tilde{H} \setminus H$ and the other endpoint in $V \setminus \tilde{H}$ (Line 24). This is equivalent to applying Algorithm 1 to $G[L]$ since $L = (V \setminus \tilde{H}) \cup (\tilde{H} \setminus H)$. Then the approximation guarantee of Algorithm 2 follows directly from Theorem 6 (with failure probability $\frac{\delta}{2}$) and Lemma 10 by using union bound. ◀

4.2 Arbitrary Order Streams

Overview of the Algorithm. Unlike random order streams, where we can identify a set $\tilde{H}$, containing the high-degree vertices H, by storing only a small number of edges from the beginning of the stream, finding high-degree vertices is more challenging in arbitrary order streams. To handle this, we employ *reservoir sampling* [47]. Specifically, we uniformly sample $\mathrm{poly}(1/\epsilon, 1/\delta)$ edges from the stream. Then, at the end of the stream, we use these sampled edges to compute a small set $\tilde{H}$ such that $\tilde{H} \supseteq H$. This approach is similar to what we used in the random order stream; however, in this case, we can only retrieve $\tilde{H}$ rather than H at the end of the stream.

We also need to estimate the number of incident edges related to high-degree vertices. To this end, we use techniques from vector sketching, particularly those developed for *frequency estimation* and *heavy hitters*. We consider the sketching techniques for heavy hitters, specifically the randomized summaries of CountMin sketch [15], corresponding to V^+ and V^-, denoted by $\mathbf{CM}[V^+]$ and $\mathbf{CM}[V^-]$, respectively. Intuitively, by the end of the stream, $\mathbf{CM}[V^+]$ and $\mathbf{CM}[V^-]$ will contain the estimates of the number of incident edges of high-degree vertices to sets V^+ and V^-, respectively. More precisely, as an edge (u, v) arrive, we increment the counters related to u in $\mathbf{CM}[V^{\mathcal{O}(v)}]$, and increment the counters related to v in $\mathbf{CM}[V^{\mathcal{O}(u)}]$, where $\mathcal{O}(v)$ and $\mathcal{O}(u)$ are respectively the predicted sign of v, u by the given ϵ-accurate oracle $\mathcal{O}$.

Note that we cannot detect H exactly by the end of stream. Instead, we have $\tilde{H}$. We use $\mathbf{CM}[V^+]$ and $\mathbf{CM}[V^-]$ to approximately compute the value of the cut $(\tilde{L}^+, \tilde{L}^-)$ on the induced subgraph $G[\tilde{L}]$, where $\tilde{L} := V \setminus \tilde{H}$. Then, using $\mathbf{CM}[V^+]$ and $\mathbf{CM}[V^-]$, we can approximately run the greedy approach for $\tilde{H}$. Unlike the random order setting where we could implement the greedy extension exactly, here we can only store the approximate values. However, we show that our estimates of the number of incident edges of high-degree vertices to sets V^+ and V^- are only off by $\mathrm{poly}(\epsilon, \delta) \cdot m$ additive terms with high probability, and they suffice to provide a strictly better than $\frac{1}{2}$-approximation for MAX-CUT on $G[\tilde{L}]$ and its extension with $\tilde{H}$. Similarly, we can use $\mathbf{CM}[V^+]$ and $\mathbf{CM}[V^-]$ to compute the size of $(\tilde{H}, \tilde{L})$ approximately too. Then, we can show that for sufficiently small value of ϵ, the best of two candidate cuts is a $(\frac{1}{2} + \Omega(\epsilon^2))$-approximation for MAX-CUT on G with probability $1 - \delta$.

Next, we prove the approximation guarantee and the space complexity of our algorithm for arbitrary order streams. The algorithm is formally given in Algorithm 3.

▶ **Theorem 11.** *Let $\epsilon \in (0, \frac{1}{2}]$ and $\delta \in (0, 1)$. Given an ϵ-accurate oracle $\mathcal{O}$ for the MAX-CUT of G, there exists a single-pass $(\frac{1}{2} + \Omega(\epsilon^2))$-approximation algorithm for estimating the MAX-CUT value of G in the insertion-only arbitrary-order streams. The algorithm uses $O(\frac{1}{\epsilon^7 \delta^3} \ln \frac{1}{\epsilon\delta})$ words of space. The approximation holds with probability at least $1 - \delta$.*

▶ **Lemma 12.** *Let $\delta \in (0, 1)$. Then $\tilde{H}$ contains all high-degree vertices H in G with probability at least $1 - \frac{\delta}{2}$.*

Proof. The proof is basically similar to that of Lemma 10. Suppose that v is a high-degree vertex in G, i.e., $\deg(v) \geq \frac{\epsilon^2 m}{c}$. For each edge e_i incident to v, define an indicator random variable

$$X_i = \begin{cases} 1, & \text{if } e_i \in F \\ 0, & \text{otherwise} \end{cases}$$

Algorithm 3 Estimating the MAX-CUT value in (insertion-only) arbitrary-order streams.

Input: Graph G as an arbitrary-order stream of edges, an ϵ-accurate oracle $\mathcal{O} \to \{-1, 1\}$
Output: Estimate of the MAX-CUT value of G

$\triangleright$ **Preprocessing phase**

Initialize $e(V^+, V^-) \leftarrow 0$.
Set $h_1, \ldots, h_k : [n] \to [w]$ be 2-wise independent hash functions, with $k = \lceil \frac{e}{\epsilon^7 \delta^3} \rceil$ and $w = \lceil \ln \frac{8\beta}{\epsilon^4 \delta^4} \rceil$ (where β is a sufficiently large universal constant).
Initialize $\mathbf{CM}[V^+]$ and $\mathbf{CM}[V^-]$ to zero.

$\triangleright$ **Streaming phase**

for each edge (u, v) in the stream **do**
  $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  if $Y_u \neq Y_v$ **then** $e(V^+, V^-) \leftarrow e(V^+, V^-) + 1$
  for each $\ell \in [k]$ **do**
    $\mathbf{CM}[V^{\mathcal{O}(u)}][\ell, h_\ell(v)] \leftarrow \mathbf{CM}[V^{\mathcal{O}(u)}][\ell, h_\ell(v)] + 1$
    $\mathbf{CM}[V^{\mathcal{O}(v)}][\ell, h_\ell(u)] \leftarrow \mathbf{CM}[V^{\mathcal{O}(v)}][\ell, h_\ell(u)] + 1$
In parallel, uniformly sample $\frac{\beta}{\delta^3 \epsilon^4}$ edges F in the stream via reservoir sampling [47].

$\triangleright$ **Postprocessing phase**

for each $v \in V$ **do**
  $f_v^+ \leftarrow \min_{\ell \in [k]} \mathbf{CM}[V^+][\ell, h_\ell(v)]$
  $f_v^- \leftarrow \min_{\ell \in [k]} \mathbf{CM}[V^-][\ell, h_\ell(v)]$
$\tilde{H} := \bigcup_{(u,v) \in F} \{u, v\}$
$\tilde{H}^+ \leftarrow \{v \in \tilde{H} : Y_v = 1\}, \tilde{H}^- \leftarrow V \setminus \tilde{H}^+$
$\tilde{e}(\tilde{L}^+, \tilde{L}^-) \leftarrow e(V^+, V^-) - \sum_{v \in \tilde{H}^+} f_v^- - \sum_{v \in \tilde{H}^-} f_v^+$
$\mathrm{ALG}_1 \leftarrow \tilde{e}(\tilde{L}^+, \tilde{L}^-) + \sum_{v \in \tilde{H}} \max\{f_v^-, f_v^+\}$
$\mathrm{ALG}_2 \leftarrow \sum_{v \in \tilde{H}} (f_v^- + f_v^+)$
return $\max\{\mathrm{ALG}_1, \mathrm{ALG}_2\}$

So, $\mathbb{E}[X_i] = \Pr[X_i = 1] = \frac{\beta}{\delta^3 \epsilon^4 m}$. We define $X := \sum_{i \in [\deg(v)]} X_i$ to denote the number of edges incident to v that are sampled by the end of the stream. Then, $\mathbb{E}[X] = \sum_{i \in [\deg(v)]} \mathbb{E}[X_i] = \frac{\beta \deg(v)}{\delta^3 \epsilon^4 m}$.

Next, we bound $\mathrm{Var}[X]$. For any $i \in [\deg(v)]$, $\mathrm{Var}[X_i] = \mathbb{E}[X_i^2] - (\mathbb{E}[X_i])^2 = \mathbb{E}[X_i] - (\mathbb{E}[X_i])^2 = \frac{\beta}{\delta^3 \epsilon^4 m} - (\frac{\beta}{\delta^3 \epsilon^4 m})^2 = \frac{\beta}{\delta^3 \epsilon^4 m} - \frac{\beta^2}{\delta^6 \epsilon^8 m^2}$. Since $(X_i)_{i \in [\deg(v)]}$ are negatively correlated, $\mathrm{Var}[X] \leq \sum_{i \in [\deg(v)]} \mathrm{Var}[X_i] = \deg(v) \cdot \left(\frac{\beta}{\delta^3 \epsilon^4 m} - \frac{\beta^2}{\delta^6 \epsilon^8 m^2} \right) \leq \frac{\beta \deg(v)}{\delta^3 \epsilon^4 m}$. Therefore, by Chebyshev's inequality,

$$\begin{aligned}
\Pr[v \notin \tilde{H}] = \Pr[X \leq 0] &= \Pr\left[X \leq \mathbb{E}[X] - \frac{\beta \deg(v)}{\delta^3 \epsilon^4 m}\right] \\
&\leq \frac{\mathrm{Var}[X] \cdot \delta^6 \epsilon^8 m^2}{\beta^2 \deg^2(v)} \\
&\leq \frac{\delta^3 \epsilon^4 m}{\beta \cdot \deg(v)} && \triangleright\ \mathrm{Var}[X] \leq \frac{\beta \deg(v)}{\delta^3 \epsilon^4 m} \\
&\leq \frac{\delta^3 \epsilon^2 c}{\beta}. && \triangleright\ \deg(v) \geq \frac{\epsilon^2 m}{c}
\end{aligned}$$

By our definition of high-degree vertices, there are at most $\frac{2m}{\epsilon^2 m / c} = \frac{2c}{\epsilon^2}$ high-degree vertices in G. By union bound, with probability at least $1 - \frac{2c}{\epsilon^2} \cdot \frac{\delta^3 \epsilon^2 c}{\beta} = 1 - \frac{2c^2 \delta^3}{\beta} \geq 1 - \frac{\delta}{2}$ (since $c = \frac{80}{\delta}$ and β sufficiently large), $\tilde{H}$ contains all high-degree vertices in G. ◀

▶ **Lemma 13.** *Let $\epsilon \in (0, \frac{1}{2}]$ and $\delta \in (0,1)$. For all vertices $v \in \tilde{H}$, (1) $f_v^+ \geq e(v, V^+)$ and $f_v^- \geq e(v, V^-)$. (2) with probability at least $1 - \frac{\delta}{4}$, $f_v^+ \leq e(v, V^+) + 2\epsilon^7\delta^3 m$ and $f_v^- \leq e(v, V^-) + 2\epsilon^7\delta^3 m$.*

Proof. (1) By Theorem 2, we have $f_v^+ \geq e(v, V^+)$ and $f_v^- \geq e(v, V^-)$ for all $v \in V$.

(2) By Theorem 2, we have $f_v^+ \leq e(v, V^+) + 2\epsilon^7\delta^3 m$ and $f_v^- \leq e(v, V^-) + 2\epsilon^7\delta^3 m$ with probability at least $1 - \frac{\epsilon^4\delta^4}{8\beta}$ for any $v \in V$. Since $|\tilde{H}| \leq \frac{2\beta}{\delta^3\epsilon^4}$, by union bound, with probability at least $1 - \frac{\epsilon^4\delta^4}{8\beta} \cdot \frac{2\beta}{\delta^3\epsilon^4} = 1 - \frac{\delta}{4}$, we have $f_v^+ \leq e(v, V^+) + 2\epsilon^7\delta^3 m$ and $f_v^- \leq e(v, V^-) + 2\epsilon^7\delta^3 m$ for all $v \in \tilde{H}$. ◀

Proof of Theorem 11. In Algorithm 3, we use CountMin sketch with $k = \lceil \frac{e}{\epsilon^7\delta^3} \rceil$ and $w = \lceil \ln \frac{8\beta}{\epsilon^4\delta^4} \rceil$ to estimate the number of incident edges of high-degree vertices to with respect to V^+ and V^-, which takes $O(\frac{1}{\epsilon^7\delta^3} \ln \frac{1}{\epsilon^4\delta^4}) = O(\frac{1}{\epsilon^7\delta^3} \ln \frac{1}{\epsilon\delta})$ words of space, by Theorem 2. Also, we use reservoir sampling to uniformly sample $O(\frac{1}{\delta^3\epsilon^4})$ edges, which takes $O(\frac{1}{\delta^3\epsilon^4})$ words of space. Therefore, the total space complexity of Algorithm 3 is $O(\frac{1}{\epsilon^7\delta^3} \ln \frac{1}{\epsilon\delta})$ words.

Next, we show the approximation guarantee of Algorithm 3. Recall that in arbitrary order streams, we cannot detect H by the end of stream, and we have $\tilde{H}$ instead. Let $A_1 := e(\tilde{L}^+, \tilde{L}^-) + \sum_{v \in \tilde{H}} \max\{e(v, \tilde{L}^-), e(v, \tilde{L}^+)\}$ denote the value of the cut $(\tilde{C}, V \setminus \tilde{C})$ obtained by running Algorithm 1 on $G[\tilde{L}]$ and then assigning the vertices in $\tilde{H}$ in a greedy manner. Let $A_2 := \sum_{v \in \tilde{H}} (e(v, \tilde{L}^-) + e(v, \tilde{L}^+))$ denote the value of the cut $(\tilde{H}, \tilde{L})$. Suppose that the best of $(\tilde{C}, V \setminus \tilde{C})$ and $(\tilde{H}, \tilde{L})$ cuts is a $(\frac{1}{2} + \frac{\epsilon^2}{16})$-approximation for the MAX-CUT value of G. Recall that $\mathrm{ALG}_1 = \tilde{e}(\tilde{L}^+, \tilde{L}^-) + \sum_{v \in \tilde{H}} \max\{f_v^-, f_v^+\}$ and $\mathrm{ALG}_2 = \sum_{v \in \tilde{H}} (f_v^- + f_v^+)$. In the following, we show that ALG_1 and ALG_2 are good approximations of A_1 and A_2, respectively. Therefore, $\max\{\mathrm{ALG}_1, \mathrm{ALG}_2\}$ returned by Algorithm 3 is also a strictly better than $\frac{1}{2}$-approximation for the MAX-CUT value of G.

Since $V^+ = \tilde{H}^+ \cup \tilde{L}^+$ and $V^- = \tilde{H}^- \cup \tilde{L}^-$, we have $e(v, V^+) = e(v, \tilde{H}^+) + e(v, \tilde{L}^+)$ and $e(v, V^-) = e(v, \tilde{H}^-) + e(v, \tilde{L}^-)$ for any vertex $v \in V$. By Lemma 13, with probability at least $1 - \frac{\delta}{4}$, for all vertices $v \in \tilde{H}$, we have

$$e(v, \tilde{H}^+) + e(v, \tilde{L}^+) \leq f_v^+ \leq e(v, \tilde{H}^+) + e(v, \tilde{L}^+) + 2\epsilon^7\delta^3 m,$$
$$e(v, \tilde{H}^-) + e(v, \tilde{L}^-) \leq f_v^- \leq e(v, \tilde{H}^-) + e(v, \tilde{L}^-) + 2\epsilon^7\delta^3 m.$$

Since $|\tilde{H}| \leq \frac{2\beta}{\delta^3\epsilon^4}$, we have $0 \leq e(v, \tilde{H}^+), e(v, \tilde{H}^-) \leq \frac{2\beta}{\delta^3\epsilon^4}$. Therefore, we have

$$e(v, \tilde{L}^+) \leq f_v^+ \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m + e(v, \tilde{L}^+),$$
$$e(v, \tilde{L}^-) \leq f_v^- \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m + e(v, \tilde{L}^-).$$

Then we have

$$0 \leq f_v^+ - e(v, \tilde{L}^+) \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m \quad \text{and} \quad 0 \leq f_v^- - e(v, \tilde{L}^-) \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m,$$

and

$$\max\{e(v, \tilde{L}^+), e(v, \tilde{L}^-)\} \leq \max\{f_v^+, f_v^-\} \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m + \max\{e(v, \tilde{L}^+), e(v, \tilde{L}^-)\}.$$

Note that

$$e(\tilde{L}^+, \tilde{L}^-) = e(V^+, V^-) - e(\tilde{H}^+, \tilde{H}^-) - \sum_{v \in \tilde{H}^+} e(v, \tilde{L}^-) - \sum_{v \in \tilde{H}^-} e(v, \tilde{L}^+),$$
$$\tilde{e}(\tilde{L}^+, \tilde{L}^-) = e(V^+, V^-) - \sum_{v \in \tilde{H}^+} f_v^- - \sum_{v \in \tilde{H}^-} f_v^+.$$

We have

$$\begin{aligned}e(\tilde{L}^+,\tilde{L}^-)-\tilde{e}(\tilde{L}^+,\tilde{L}^-)&=\sum_{v\in\tilde{H}^+}(f_v^- - e(v,\tilde{L}^-))+\sum_{v\in\tilde{H}^-}(f_v^+ - e(v,\tilde{L}^+))-e(\tilde{H}^+,\tilde{H}^-)\\&\le\frac{2\beta}{\delta^3\epsilon^4}\cdot\left(\frac{2\beta}{\delta^3\epsilon^4}+2\epsilon^7\delta^3 m\right)+\frac{2\beta}{\delta^3\epsilon^4}\cdot\left(\frac{2\beta}{\delta^3\epsilon^4}+2\epsilon^7\delta^3 m\right)-0\\&=\frac{8\beta^2}{\delta^6\epsilon^8}+8\beta\epsilon^3 m.\end{aligned}$$

Therefore,

$$\begin{aligned}A_1-\mathrm{ALG}_1&=(e(\tilde{L}^+,\tilde{L}^-)-\tilde{e}(\tilde{L}^+,\tilde{L}^-))+\sum_{v\in\tilde{H}}(\max\{e(v,\tilde{L}^-),e(v,\tilde{L}^+)\}-\max\{f_v^-,f_v^+\})\\&\le\left(\frac{8\beta^2}{\delta^6\epsilon^8}+8\beta\epsilon^3 m\right)+\frac{2\beta}{\delta^3\epsilon^4}\cdot 0=\frac{8\beta^2}{\delta^6\epsilon^8}+8\beta\epsilon^3 m.\end{aligned}$$

So, $\mathrm{ALG}_1 \ge A_1 - (\frac{8\beta^2}{\delta^6\epsilon^8}+8\beta\epsilon^3 m) = A_1 - \Theta(\epsilon^3 m)$.

Similarly,

$$A_2-\mathrm{ALG}_2=\sum_{v\in\tilde{H}}((e(v,\tilde{L}^-)-f_v^-)+(e(v,\tilde{L}^+)-f_v^+))\le\frac{2\beta}{\delta^3\epsilon^4}\cdot(0+0)=0.$$

So, $\mathrm{ALG}_2 \ge A_2$.

Since we assume that $\max\{A_1, A_2\}$ is a $(\frac{1}{2}+\frac{\epsilon^2}{16})$-approximation for the MAX-CUT value of G, we have $\mathrm{ALG} := \max\{\mathrm{ALG}_1, \mathrm{ALG}_2\} \ge \max\{A_1, A_2\} - \Theta(\epsilon^3 m) \ge (\frac{1}{2}+\frac{\epsilon^2}{16})\cdot \mathrm{OPT} - \Theta(\epsilon^3\cdot \mathrm{OPT}) = (\frac{1}{2}+\Omega(\epsilon^2))\cdot\mathrm{OPT}$.

Finally, it remains to show that the best of $(\tilde{C}, V\setminus\tilde{C})$ and $(\tilde{H},\tilde{L})$ cuts is a $(\frac{1}{2}+\frac{\epsilon^2}{16})$-approximation for the MAX-CUT value of G. This directly follows from Theorem 6 (with failure probability $\frac{\delta}{4}$), by substituting H and L with $\tilde{H}$ and $\tilde{L}$, respectively. Together with Lemma 12, Lemma 13, and applying union bound, this concludes the proof. ◀

4.3 Extension to Dynamic Streams

Overview of the Algorithm. Our algorithm in dynamic streams is basically similar to Algorithm 3. The only difference is that to compute a small set $\tilde{H}$ at the end of the stream such that $\tilde{H}\supseteq H$, instead of reservoir sampling, we need to use ℓ_0-sampling [28].

Next, we prove the approximation guarantee and the space complexity of our algorithm for dynamic streams. The algorithm is formally given in Algorithm 4.

▶ **Theorem 14.** *Let $\epsilon\in(0,\frac{1}{2}]$ and $\delta\in(0,1)$. Given an ϵ-accurate oracle $\mathcal{O}$ for the MAX-CUT of G, there exists a single-pass $(\frac{1}{2}+\Omega(\epsilon^2))$-approximation algorithm for estimating the MAX-CUT value of G in the dynamic streams. The algorithm uses $O(\frac{\log^2 n}{\epsilon^7\delta^3}\log\frac{1}{\epsilon\delta})$ bits of space. The approximation holds with probability at least $1-\delta$.*

Note that Theorem 1 follows from Theorem 11 and Theorem 14 by setting $\delta = 1/3$.

Proof Sketch of Theorem 14. The proof follows a similar structure to that of Theorem 11. First, we show that ℓ_0-sampling in dynamic streams can effectively replace reservoir sampling in insertion-only streams, allowing us to retrieve a set $\tilde{H}\supseteq H$ with high probability by the end of the stream.

Algorithm 4 Estimating the MAX-CUT value in dynamic streams.

Input: Graph G as a dynamic stream of edges, an ϵ-accurate oracle $\mathcal{O} \to \{-1, 1\}$
Output: Estimate of the MAX-CUT value of G

▷ **Preprocessing phase**
Initialize $e(V^+, V^-) \leftarrow 0$.
Set $h_1, \ldots, h_k : [n] \to [w]$ be 2-wise independent hash functions, with $k = \lceil \frac{e}{\epsilon^7 \delta^3} \rceil$ and $w = \lceil \ln \frac{8\beta}{\epsilon^4 \delta^4} \rceil$ (where β is a sufficiently large universal constant).
Initialize $\mathbf{CM}[V^+]$ and $\mathbf{CM}[V^-]$ to zero.
▷ **Streaming phase**
for each item $((u, v), \Delta_{(u,v)})$ in the stream **do**
  $Y_u = \mathcal{O}(u), Y_v = \mathcal{O}(v)$
  if $Y_u \neq Y_v$ **then** $e(V^+, V^-) \leftarrow e(V^+, V^-) + 1$
  for each $\ell \in [k]$ **do**
    $\mathbf{CM}[V^{\mathcal{O}(u)}][\ell, h_\ell(v)] \leftarrow \mathbf{CM}[V^{\mathcal{O}(u)}][\ell, h_\ell(v)] + \Delta_{(u,v)}$
    $\mathbf{CM}[V^{\mathcal{O}(v)}][\ell, h_\ell(u)] \leftarrow \mathbf{CM}[V^{\mathcal{O}(v)}][\ell, h_\ell(u)] + \Delta_{(u,v)}$
In parallel, maintain $\frac{\beta}{\delta^3 \epsilon^4}$ ℓ_0-samplers (with failure probability $\delta' = \frac{\delta^4 \epsilon^4}{8\beta}$) to obtain $\frac{\beta}{\delta^3 \epsilon^4}$ different edges F.
▷ **Postprocessing phase**
for each $v \in V$ **do**
  $f_v^+ \leftarrow \min_{\ell \in [k]} \mathbf{CM}[V^+][\ell, h_\ell(v)]$
  $f_v^- \leftarrow \min_{\ell \in [k]} \mathbf{CM}[V^-][\ell, h_\ell(v)]$
$\tilde{H} := \bigcup_{(u,v) \in F} \{u, v\}$
$\tilde{H}^+ \leftarrow \{v \in \tilde{H} : Y_v = 1\}, \tilde{H}^- \leftarrow V \setminus \tilde{H}^+$
$\tilde{e}(\tilde{L}^+, \tilde{L}^-) \leftarrow e(V^+, V^-) - \sum_{v \in \tilde{H}^+} f_v^- - \sum_{v \in \tilde{H}^-} f_v^+$
$\mathrm{ALG}_1 \leftarrow \tilde{e}(\tilde{L}^+, \tilde{L}^-) + \sum_{v \in \tilde{H}} \max\{f_v^-, f_v^+\}$
$\mathrm{ALG}_2 \leftarrow \sum_{v \in \tilde{H}} (f_v^- + f_v^+)$
return $\max\{\mathrm{ALG}_1, \mathrm{ALG}_2\}$

Next, we show that for all vertices $v \in \tilde{H}$, our estimates of the number of incident edges from v to the sets V^+ and V^- has additive error at most $\mathrm{poly}(\epsilon, \delta) \cdot m$, with high probability.

Finally, using arguments similar to those in the proof of Theorem 11, we can show that Algorithm 4 provides a $(\frac{1}{2} + \Omega(\epsilon^2))$-approximation for the MAX-CUT value of G with probability $1 - \delta$. For completeness, we provide the detailed proof in Section A.

5 Conclusion

We present the first learning-augmented algorithm for estimating the value of MAX-CUT in the streaming setting by leveraging ϵ-accurate predictions. Specifically, we provide a single-pass streaming algorithm that achieves a $(1/2 + \Omega(\epsilon^2))$-approximation for estimating the MAX-CUT value of a graph in insertion-only (respectively, fully dynamic) streams using $\mathrm{poly}(1/\epsilon)$ (respectively, $\mathrm{poly}(1/\epsilon, \log n)$) words of space. This result contrasts with the lower bound in the classical streaming setting (without predictions), where any (randomized) single-pass streaming algorithm that achieves an approximation ratio of at least $1/2 + \epsilon$ requires $\Omega(n/2^{\mathrm{poly}(1/\epsilon)})$ space.

Our work leaves several questions for further research. For example, it would be interesting to extend our algorithm from unweighted to weighted graphs. Currently, our algorithm for handling the low-degree part does not seem to work for weighted graphs, as an edge with a heavy weight but incorrectly predicted signs can significantly affect our estimator. Another open question is to establish lower bounds on the trade-offs between approximation ratio and space complexity when the streaming algorithm is provided with predictions.

References

1. Anders Aamand, Justin Chen, Huy Nguyen, Sandeep Silwal, and Ali Vakilian. Improved frequency estimation algorithms with and without predictions. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
2. Anders Aamand, Piotr Indyk, and Ali Vakilian. frequency estimation algorithms under zipfian distribution. *arXiv preprint*, 2019. `arXiv:1908.05198`.
3. Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems (PODS)*, pages 5–14, 2012. `doi:10.1145/2213556.2213560`.
4. Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc Renault. Online computation with untrusted advice. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151 of *LIPIcs*, pages 52:1–52:15, 2020. `doi:10.4230/LIPICS.ITCS.2020.52`.
5. Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. *ACM Trans. Algorithms*, 19(2):19:1–19:34, 2023. `doi:10.1145/3582689`.
6. Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
7. Etienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
8. Siddhartha Banerjee, Vincent Cohen-Addad, Anupam Gupta, and Zhouzi Li. Graph searching with predictions. In *14th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 251 of *LIPIcs*, pages 12:1–12:24, 2023. `doi:10.4230/LIPICS.ITCS.2023.12`.
9. Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing - Problems and Techniques*. Springer, 2022. `doi:10.1007/978-981-16-8622-1`.
10. Jan van den Brand, Sebastian Forster, Yasamin Nazari, and Adam Polak. On dynamic graph algorithms with predictions. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3534–3557, 2024. `doi:10.1137/1.9781611977912.126`.
11. Vladimir Braverman, Prathamesh Dharangutte, Vihan Shah, and Chen Wang. Learning-augmented maximum independent set. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 317 of *LIPIcs*, pages 24:1–24:18, 2024. `doi:10.4230/LIPICS.APPROX/RANDOM.2024.24`.
12. Justin Chen, Sandeep Silwal, Ali Vakilian, and Fred Zhang. Faster fundamental graph algorithms via learned predictions. In *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 3583–3602, 2022. URL: `https://proceedings.mlr.press/v162/chen22v.html`.
13. Justin Y. Chen, Talya Eden, Piotr Indyk, Honghao Lin, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, Tal Wagner, David P. Woodruff, and Michael Zhang. Triangle and four cycle counting with predictions in graph streams. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022. URL: `https://openreview.net/forum?id=8in_5gN9I0`.
14. Vincent Cohen-Addad, Tommaso d'Orsi, Anupam Gupta, Euiwoong Lee, and Debmalya Panigrahi. Learning-augmented approximation algorithms for maximum cut and related problems. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

15 Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005. doi:10.1016/J.JALGOR.2003.12.001.

16 Sami Davies, Benjamin Moseley, Sergei Vassilvitskii, and Yuyan Wang. Predictive flows for faster ford-fulkerson. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 7231–7248, 2023. URL: https://proceedings.mlr.press/v202/davies23b.html.

17 Adela F DePavia, Erasmo Tani, and Ali Vakilian. Learning-based algorithms for graph searching problems. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 238 of *Proceedings of Machine Learning Research*, pages 928–936, 2024. URL: https://proceedings.mlr.press/v238/depavia24a.html.

18 Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 10393–10406, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/5616060fb8ae85d93f334e7267307664-Abstract.html.

19 Talya Eden, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, and Tal Wagner. Learning-based support estimation in sublinear time. In *9th International Conference on Learning Representations (ICLR)*, 2021. URL: https://openreview.net/forum?id=tilovEHA3YS.

20 Paolo Ferragina and Giorgio Vinciguerra. The pgm-index: a fully-dynamic compressed learned index with provable worst-case bounds. *Proc. VLDB Endow.*, 13(8):1162–1175, 2020. doi:10.14778/3389133.3389135.

21 Suprovat Ghoshal, Konstantin Makarychev, and Yury Makarychev. Constraint satisfaction problems with advice. In *Proceedings of the 2025 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2025.

22 Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995. doi:10.1145/227683.227684.

23 Monika Henzinger, Barna Saha, Martin P Seybold, and Christopher Ye. On the complexity of algorithms with predictions for dynamic graph problems. In *15th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 287 of *LIPIcs*, pages 62:1–62:25, 2024. doi:10.4230/LIPICS.ITCS.2024.62.

24 Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *7th International Conference on Learning Representations (ICLR)*, 2019.

25 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Online knapsack with frequency predictions. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2733–2743, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/161c5c5ad51fcc884157890511b3c8b0-Abstract.html.

26 Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 7400–7410, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/1625abb8e458a79765c62009235e9d5b-Abstract.html.

27 Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P Woodruff. Learning-augmented data stream algorithms. In *8th International Conference on Learning Representations (ICLR)*, 2020.

28 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS)*, pages 49–58, 2011.

29 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating max-cut. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1263–1282, 2015. doi:10.1137/1.9781611973730.84.

30 Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. (1+ $\omega(1)$)-approximation to max-cut requires linear space. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1703–1722, 2017. doi:10.1137/1.9781611974782.112.

31 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)*, pages 277–288, 2019. doi:10.1145/3313276.3316364.

32 Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1814–1833, 2020. doi:10.1137/1.9781611975994.111.

33 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

34 Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 367–376, 2015. doi:10.1145/2688073.2688093.

35 Ravi Kumar, Manish Purohit, and Zoya Svitkina. Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 9684–9693, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html.

36 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1859–1877, 2020. doi:10.1137/1.9781611975994.114.

37 Silvio Lattanzi, Ola Svensson, and Sergei Vassilvitskii. Speeding up bellman ford via minimum violation permutations. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 18584–18598, 2023. URL: https://proceedings.mlr.press/v202/lattanzi23a.html.

38 Yi Li, Honghao Lin, Simin Liu, Ali Vakilian, and David P. Woodruff. Learning the positions in countsketch. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

39 Honghao Lin, Tian Luo, and David P. Woodruff. Learning augmented binary search trees. In *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 13431–13440, 2022. URL: https://proceedings.mlr.press/v162/lin22f.html.

40 Quanquan C Liu and Vaidehi Srinivas. The predicted-deletion dynamic model: Taking advantage of ml predictions, for free. *arXiv preprint*, 2023. doi:10.48550/arXiv.2307.08890.

41 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25, 2021. doi:10.1145/3447579.

42 Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 288–294, 2007. doi:10.1145/1250910.1250952.

43 Michael Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 462–471, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/0f49c89d1e7298bb9930789c8ed59d48-Abstract.html.

44 Atsuki Sato and Yusuke Matsui. Fast partitioned learned bloom filter. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

45 Nicholas Schiefer, Justin Y Chen, Piotr Indyk, Shyam Narayanan, Sandeep Silwal, and Tal Wagner. Learned interpolation for better streaming quantile approximation with worst-case guarantees. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA)*, pages 87–97, 2023. doi:10.1137/1.9781611977714.8.

46 Kapil Vaidya, Eric Knorr, Michael Mitzenmacher, and Tim Kraska. Partitioned learned bloom filters. In *9th International Conference on Learning Representations (ICLR)*, 2021.

47 Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985. doi:10.1145/3147.3165.

A Proof of Theorem 14

▶ **Lemma 15.** *Let $\epsilon \in (0, \frac{1}{2}]$ and $\delta \in (0,1)$. With probability at least $1 - \frac{\delta}{4}$, Line 10 of Algorithm 4 returns a set F which contains $\frac{\beta}{\delta^3\epsilon^4}$ different edges.*

Proof. The error is introduced by the ℓ_0-samplers and the probability that sampling $\frac{\beta}{\delta^3\epsilon^4}$ uniform random edges in parallel does not yield $\frac{\beta}{\delta^3\epsilon^4}$ different edges.

By Theorem 4, the ℓ_0-sampler fails with probability at most δ'. Since we maintain $\frac{\beta}{\delta^3\epsilon^4}$ ℓ_0-samplers, by union bound, the failure probability introduced by ℓ_0-samplers is at most $\frac{\beta}{\delta^3\epsilon^4} \cdot \delta'$.

Suppose that all the ℓ_0-samplers succeed, i.e., $|F| = \frac{\beta}{\delta^3\epsilon^4}$. Now we bound the probability that there exist two identical edges in F. Consider two fixed edges $e_1, e_2 \in F$. Then $\Pr[e_1 = e_2] = \frac{1}{m}$. By union bound, the probability that there exists at least one pair of identical edges is at most $\binom{\beta/\delta^3\epsilon^4}{2} \cdot \frac{1}{m} \leq \frac{\beta^2}{\delta^6\epsilon^8 m}$.

Therefore, the probability that F contains $\frac{\beta}{\delta^3\epsilon^4}$ different edges is at least $1 - \frac{\beta}{\delta^3\epsilon^4} \cdot \delta' - \frac{\beta^2}{\delta^6\epsilon^8 m} \geq 1 - \frac{\delta}{8} - \frac{\delta}{8} \geq 1 - \frac{\delta}{4}$ (since $m = \Omega(\epsilon^{-11}\delta^{-7})$). ◀

▶ **Lemma 16.** *Let $\delta \in (0,1)$. By the end of the stream, $\tilde{H}$ contains all high-degree vertices in G with probability at least $1 - \frac{\delta}{4}$.*

Proof. Suppose that v is a high-degree vertex in G, i.e., $\deg(v) \geq \frac{\epsilon^2 m}{c}$. For each edge e_i incident to v, define an indicator random variable

$$X_i = \begin{cases} 1, & \text{if } e_i \in F \\ 0, & \text{otherwise} \end{cases}$$

So, $\mathbb{E}[X_i] = \Pr[X_i = 1] = 1 - (1 - \frac{1}{m})^{\beta/\delta^3\epsilon^4} \geq 1 - \exp\left(-\frac{\beta}{\delta^3\epsilon^4 m}\right) \geq \frac{\beta}{\delta^3\epsilon^4 m} - \frac{\beta^2}{2\delta^6\epsilon^8 m^2} \geq \frac{\beta}{5\delta^3\epsilon^4 m}$ for sufficiently large m. We define $X := \sum_{i \in [\deg(v)]} X_i$ to denote the number of edges incident to v that are sampled by the end of the stream. Then, $\mathbb{E}[X] = \sum_{i \in [\deg(v)]} \mathbb{E}[X_i] \geq \frac{\beta \deg(v)}{5\delta^3\epsilon^4 m}$.

Next, we bound $\mathrm{Var}[X]$. For any $i \in [\deg(v)]$, $\mathrm{Var}[X_i] = \mathbb{E}[X_i^2] - (\mathbb{E}[X_i])^2 = \mathbb{E}[X_i] - (\mathbb{E}[X_i])^2 \leq \frac{\beta}{5\delta^3\epsilon^4 m} - (\frac{\beta}{5\delta^3\epsilon^4 m})^2 = \frac{\beta}{5\delta^3\epsilon^4 m} - \frac{\beta^2}{25\delta^6\epsilon^8 m^2}$. Since $(X_i)_{i \in [\deg(v)]}$ are negatively correlated, $\mathrm{Var}[X] \leq \sum_{i \in [\deg(v)]} \mathrm{Var}[X_i] = \deg(v) \cdot \left(\frac{\beta}{5\delta^3\epsilon^4 m} - \frac{\beta^2}{25\delta^6\epsilon^8 m^2}\right) \leq \frac{\beta \deg(v)}{5\delta^3\epsilon^4 m}$. Therefore, by Chebyshev's inequality,

$$
\begin{aligned}
\Pr[v \notin \tilde{H}] = \Pr[X \le 0] &\le \Pr\left[X \le \mathbb{E}[X] - \frac{\beta \deg(v)}{5\delta^3\epsilon^4 m}\right] \\
&\le \frac{\mathrm{Var}[X] \cdot 25\delta^6\epsilon^8 m^2}{\beta^2 \deg^2(v)} \\
&\le \frac{5\delta^3\epsilon^4 m}{\beta \cdot \deg(v)} && \triangleright\ \mathrm{Var}[X] \le \frac{\beta \deg(v)}{5\delta^3\epsilon^4 m} \\
&\le \frac{5\delta^3\epsilon^2 c}{\beta}. && \triangleright\ \deg(v) \ge \frac{\epsilon^2 m}{c}
\end{aligned}
$$

By our definition of high-degree vertices, there are at most $\frac{2m}{\epsilon^2 m/c} = \frac{2c}{\epsilon^2}$ high-degree vertices in G. By union bound, with probability at least $1 - \frac{2c}{\epsilon^2} \cdot \frac{5\delta^3\epsilon^2 c}{\beta} = 1 - \frac{10c^2\delta^3}{\beta} \ge 1 - \frac{\delta}{4}$ (since $c = \frac{80}{\delta}$ and β sufficiently large), $\tilde{H}$ contains all high-degree vertices in G. ◀

▶ **Lemma 17.** *Let $\epsilon \in (0, \frac{1}{2}]$ and $\delta \in (0,1)$. For all vertices $v \in \tilde{H}$, (1) $f_v^+ \ge e(v, V^+)$ and $f_v^- \ge e(v, V^-)$. (2) with probability at least $1 - \frac{\delta}{4}$, $f_v^+ \le e(v, V^+) + 2\epsilon^7\delta^3 m$ and $f_v^- \le e(v, V^-) + 2\epsilon^7\delta^3 m$.*

Proof. (1) By Theorem 2, we have $f_v^+ \ge e(v, V^+)$ and $f_v^- \ge e(v, V^-)$ for all $v \in V$.

(2) By Theorem 2, we have $f_v^+ \le e(v, V^+) + 2\epsilon^7\delta^3 m$ and $f_v^- \le e(v, V^-) + 2\epsilon^7\delta^3 m$ with probability at least $1 - \frac{\epsilon^4\delta^4}{8\beta}$ for any $v \in V$. Since $|\tilde{H}| \le \frac{2\beta}{\delta^3\epsilon^4}$, by union bound, with probability at least $1 - \frac{\epsilon^4\delta^4}{8\beta} \cdot \frac{2\beta}{\delta^3\epsilon^4} = 1 - \frac{\delta}{4}$, we have $f_v^+ \le e(v, V^+) + 2\epsilon^7\delta^3 m$ and $f_v^- \le e(v, V^-) + 2\epsilon^7\delta^3 m$ for all $v \in \tilde{H}$. ◀

Proof of Theorem 14. In Algorithm 4, we use CountMin sketch with $k = \lceil \frac{e}{\epsilon^7\delta^3} \rceil$ and $w = \lceil \ln \frac{8\beta}{\epsilon^4\delta^4} \rceil$ to estimate the number of incident edges of high-degree vertices to with respect to V^+ and V^-, which takes $O(\frac{1}{\epsilon^7\delta^3} \ln \frac{1}{\epsilon^4\delta^4}) = O(\frac{1}{\epsilon^7\delta^3} \ln \frac{1}{\epsilon\delta})$ words of space, by Theorem 2. Also, we maintain $O(\frac{1}{\delta^3\epsilon^4})$ ℓ_0-samplers, which takes $O(\frac{\log^2 n}{\delta^3\epsilon^4} \log \frac{1}{\delta^4\epsilon^4}) = O(\frac{\log^2 n}{\delta^3\epsilon^4} \log \frac{1}{\delta\epsilon})$ bits of space, by Theorem 4. Therefore, the total space complexity of Algorithm 4 is $O(\frac{\log^2 n}{\epsilon^7\delta^3} \log \frac{1}{\epsilon\delta})$ bits.

Next, we show the approximation guarantee of Algorithm 4. Similar to arbitrary order streams, we cannot detect H by the end of stream. Instead, we have $\tilde{H}$. Let $A_1 := e(\tilde{L}^+, \tilde{L}^-) + \sum_{v \in \tilde{H}} \max\{e(v, \tilde{L}^-), e(v, \tilde{L}^+)\}$ denote the value of the cut $(\tilde{C}, V \setminus \tilde{C})$ obtained by running Algorithm 1 on $G[\tilde{L}]$ and then assigning the vertices in $\tilde{H}$ in a greedy manner. Let $A_2 := \sum_{v \in \tilde{H}}(e(v, \tilde{L}^-) + e(v, \tilde{L}^+))$ denote the value of the cut $(\tilde{H}, \tilde{L})$. Suppose that the best of $(\tilde{C}, V \setminus \tilde{C})$ and $(\tilde{H}, \tilde{L})$ cuts is a $(\frac{1}{2} + \frac{\epsilon^2}{16})$-approximation for the MAX-CUT value of G. In the following, we show that ALG_1 and ALG_2 are good approximations of A_1 and A_2, respectively. Therefore, $\max\{\mathrm{ALG}_1, \mathrm{ALG}_2\}$ returned by Algorithm 4 is also a strictly better than $\frac{1}{2}$-approximation for the MAX-CUT value of G.

Since $V^+ = \tilde{H}^+ \cup \tilde{L}^+$ and $V^- = \tilde{H}^- \cup \tilde{L}^-$, we have $e(v, V^+) = e(v, \tilde{H}^+) + e(v, \tilde{L}^+)$ and $e(v, V^-) = e(v, \tilde{H}^-) + e(v, \tilde{L}^-)$ for any vertex $v \in V$. By Lemma 17, with probability at least $1 - \frac{\delta}{4}$, for all vertices $v \in \tilde{H}$, we have

$$
\begin{aligned}
&e(v, \tilde{H}^+) + e(v, \tilde{L}^+) \le f_v^+ \le e(v, \tilde{H}^+) + e(v, \tilde{L}^+) + 2\epsilon^7\delta^3 m, \\
&e(v, \tilde{H}^-) + e(v, \tilde{L}^-) \le f_v^- \le e(v, \tilde{H}^-) + e(v, \tilde{L}^-) + 2\epsilon^7\delta^3 m.
\end{aligned}
$$

Since $|\tilde{H}| \leq \frac{2\beta}{\delta^3\epsilon^4}$, we have $0 \leq e(v, \tilde{H}^+), e(v, \tilde{H}^-) \leq \frac{2\beta}{\delta^3\epsilon^4}$. Therefore, we have

$$e(v, \tilde{L}^+) \leq f_v^+ \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m + e(v, \tilde{L}^+),$$
$$e(v, \tilde{L}^-) \leq f_v^- \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m + e(v, \tilde{L}^-).$$

Then we have

$$0 \leq f_v^+ - e(v, \tilde{L}^+) \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m \quad \text{and} \quad 0 \leq f_v^- - e(v, \tilde{L}^-) \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m,$$

and

$$\max\{e(v, \tilde{L}^+), e(v, \tilde{L}^-)\} \leq \max\{f_v^+, f_v^-\} \leq \frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m + \max\{e(v, \tilde{L}^+), e(v, \tilde{L}^-)\}.$$

Note that

$$e(\tilde{L}^+, \tilde{L}^-) = e(V^+, V^-) - e(\tilde{H}^+, \tilde{H}^-) - \sum_{v \in \tilde{H}^+} e(v, \tilde{L}^-) - \sum_{v \in \tilde{H}^-} e(v, \tilde{L}^+),$$
$$\tilde{e}(\tilde{L}^+, \tilde{L}^-) = e(V^+, V^-) - \sum_{v \in \tilde{H}^+} f_v^- - \sum_{v \in \tilde{H}^-} f_v^+.$$

We have

$$\begin{aligned} e(\tilde{L}^+, \tilde{L}^-) - \tilde{e}(\tilde{L}^+, \tilde{L}^-) &= \sum_{v \in \tilde{H}^+} (f_v^- - e(v, \tilde{L}^-)) + \sum_{v \in \tilde{H}^-} (f_v^+ - e(v, \tilde{L}^+)) - e(\tilde{H}^+, \tilde{H}^-) \\ &\leq \frac{2\beta}{\delta^3\epsilon^4} \cdot \left(\frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m \right) + \frac{2\beta}{\delta^3\epsilon^4} \cdot \left(\frac{2\beta}{\delta^3\epsilon^4} + 2\epsilon^7\delta^3 m \right) - 0 \\ &= \frac{8\beta^2}{\delta^6\epsilon^8} + 8\beta\epsilon^3 m. \end{aligned}$$

Therefore,

$$\begin{aligned} A_1 - \mathrm{ALG}_1 &= (e(\tilde{L}^+, \tilde{L}^-) - \tilde{e}(\tilde{L}^+, \tilde{L}^-)) + \sum_{v \in \tilde{H}} (\max\{e(v, \tilde{L}^-), e(v, \tilde{L}^+)\} - \max\{f_v^-, f_v^+\}) \\ &\leq \left(\frac{8\beta^2}{\delta^6\epsilon^8} + 8\beta\epsilon^3 m \right) + \frac{2\beta}{\delta^3\epsilon^4} \cdot 0 = \frac{8\beta^2}{\delta^6\epsilon^8} + 8\beta\epsilon^3 m. \end{aligned}$$

So, $\mathrm{ALG}_1 \geq A_1 - (\frac{8\beta^2}{\delta^6\epsilon^8} + 8\beta\epsilon^3 m) = A_1 - \Theta(\epsilon^3 m)$.

Similarly,

$$A_2 - \mathrm{ALG}_2 = \sum_{v \in \tilde{H}} ((e(v, \tilde{L}^-) - f_v^-) + (e(v, \tilde{L}^+) - f_v^+)) \leq \frac{2\beta}{\delta^3\epsilon^4} \cdot (0 + 0) = 0.$$

So, $\mathrm{ALG}_2 \geq A_2$.

Since we assume that $\max\{A_1, A_2\}$ is a $(\frac{1}{2} + \frac{\epsilon^2}{16})$-approximation for the MAX-CUT value of G, we have $\mathrm{ALG} := \max\{\mathrm{ALG}_1, \mathrm{ALG}_2\} \geq \max\{A_1, A_2\} - \Theta(\epsilon^3 m) \geq (\frac{1}{2} + \frac{\epsilon^2}{16}) \cdot \mathrm{OPT} - \Theta(\epsilon^3 \cdot \mathrm{OPT}) = (\frac{1}{2} + \Omega(\epsilon^2)) \cdot \mathrm{OPT}$.

Finally, it remains to show that the best of $(\tilde{C}, V \setminus \tilde{C})$ and $(\tilde{H}, \tilde{L})$ cuts is a $(\frac{1}{2} + \frac{\epsilon^2}{16})$-approximation for the MAX-CUT value of G. This directly follows from Theorem 6 (with failure probability $\frac{\delta}{4}$), by substituting H and L with $\tilde{H}$ and $\tilde{L}$, respectively. Together with Lemma 15, Lemma 16, Lemma 17, and applying union bound, this concludes the proof. ◀

B Constant Query Complexity in Random Order Streams

In this section, we show that in the case of random-order streams, it suffices to query the ϵ-accurate oracle $\mathcal{O}$ for the labels of a constant number of vertices. For simplicity, we provide only a sketch of the main ideas and omit the formal proof. We will utilize the following lemma.

▶ **Lemma 18** (Lemma 2.2 in [9]). *Let $a, b \in \mathbb{R}$ with $a < b$, $n \in \mathbb{N}$ and $x \in [a, b]^n$. Let $\Sigma = \sum_{i=1}^{n} x_i$. For any $\eta, \delta \in (0, 1)$, there exists an algorithm which samples a set T of $t = O(\eta^{-2} \log \delta^{-1})$ indices from $[n]$ and returns an estimate $\tilde{\Sigma} = \frac{n}{t} \sum_{i \in T} x_i$ such that $|\Sigma - \tilde{\Sigma}| \le \eta(b - a)n$ with probability at least $1 - \delta$.*

Similar to Algorithm 2, we store the first $\mathrm{poly}(1/\epsilon, 1/\delta)$ edges in the random order stream and use them to identify a set $\tilde{H}$ of size $\mathrm{poly}(1/\epsilon, 1/\delta)$ that contains all high-degree vertices H with high probability as in Lemma 10. For this part, we do not need any information on the labels of vertices provided by the oracle $\mathcal{O}$. Recall that the algorithm considers two candidate cuts and returns the one with the larger size. Let $\tilde{L} := V \setminus \tilde{H}$ and $\tilde{S} := \tilde{H} \setminus H$. The first candidate is obtained by performing the greedy extension of $(\tilde{L}^+ \cup \tilde{S}^+, \tilde{L}^- \cup \tilde{S}^-)$ using H. The second candidate is simply the cut (H, L). Formally, the sizes of these two cuts are given as follows:

$$\begin{aligned}
\mathrm{ALG}_1 &= e(L^+, L^-) + \sum_{v \in H} \max\{e(v, L^-), e(v, L^+)\} \\
&= e(\tilde{L}^+, \tilde{L}^-) + e(\tilde{S}^+, \tilde{S}^-) + \sum_{v \in \tilde{S}^+} e(v, \tilde{L}^-) + \sum_{v \in \tilde{S}^-} e(v, \tilde{L}^+) + \sum_{v \in H} \max\{e(v, L^-), e(v, L^+)\}, \\
\mathrm{ALG}_2 &= e(H, L) = \sum_{v \in H} e(v, L) = \sum_{v \in H} (e(v, \tilde{L}) + e(v, \tilde{S})).
\end{aligned}$$

Observe that, to compute the second cut size ALG_2, there is no need to query the oracle $\mathcal{O}$. It suffices to count $e(v, \tilde{L})$ for each vertex $v \in \tilde{H}$ using the remaining edges (after the first $\mathrm{poly}(1/\epsilon, 1/\delta)$ edges) during the stream. At the end of the stream, we can retrieve H from $\tilde{H}$ and compute ALG_2 exactly.

Next, we focus on estimating ALG_1 by querying the oracle $\mathcal{O}$ a constant number of times. Specifically, we decompose ALG_1 into four parts and estimate each part respectively.

- **(I)** $e(\tilde{L}^+, \tilde{L}^-)$. During the stream (after the first $\mathrm{poly}(1/\epsilon, 1/\delta)$ edges), we employ reservoir sampling to sample $O(\epsilon^{-4} \log \delta^{-1})$ edges T uniformly at random from $E(G[\tilde{L}])$, the set of edges with both endpoints in $\tilde{L}$. Let $U := \bigcup_{(u,v) \in T} \{u, v\}$. By Lemma 18, we query $\mathcal{O}$ for the vertices in U and use $\frac{e(G[\tilde{L}])}{|T|} \cdot e(U^+, U^-)$ to estimate $e(\tilde{L}^+, \tilde{L}^-)$, with an additive error of $\epsilon^2 \cdot e(G[\tilde{L}])$.
- **(II)** $e(\tilde{S}^+, \tilde{S}^-)$. Since we store all edges with both endpoints in $\tilde{H}$ during the stream and $|\tilde{H}| = \mathrm{poly}(1/\epsilon, 1/\delta)$, we can compute $e(\tilde{S}^+, \tilde{S}^-)$ exactly by querying $\mathcal{O}$ a constant number of times.
- **(III)** $\sum_{v \in \tilde{S}^+} e(v, \tilde{L}^-) + \sum_{v \in \tilde{S}^-} e(v, \tilde{L}^+)$. For each vertex $v \in \tilde{H}$, we use reservoir sampling to sample $O(\epsilon^{-4} \log \delta^{-1})$ edges T_v uniformly at random from $E(v, \tilde{L})$, the set of edges with one endpoint at v and the other in $\tilde{L}$. Let $U_v := \bigcup_{(u,v) \in T_v} \{u\}$. By Lemma 18, we query $\mathcal{O}$ for the vertices in U_v and use $\frac{e(v, \tilde{L})}{|T_v|} \cdot e(v, U_v^+)$ to estimate $e(v, \tilde{L}^+)$ and use $\frac{e(v, \tilde{L})}{|T_v|} \cdot e(v, U_v^-)$ to estimate $e(v, \tilde{L}^-)$, both with an additive error of $\epsilon^2 \cdot e(v, \tilde{L})$. Therefore, we can estimate $\sum_{v \in \tilde{S}^+} e(v, \tilde{L}^-) + \sum_{v \in \tilde{S}^-} e(v, \tilde{L}^+)$ with an additive error of $\epsilon^2 \cdot \sum_{v \in \tilde{S}} e(v, \tilde{L}) = \epsilon^2 \cdot e(\tilde{S}, \tilde{L})$.

(IV) $\sum_{v \in H} \max\{e(v, L^-), e(v, L^+)\}$. Since $L^- = \tilde{S}^- \cup \tilde{L}^-$ and $L^+ = \tilde{S}^+ \cup \tilde{L}^+$, we have $e(v, L^-) = e(v, \tilde{S}^-) + e(v, \tilde{L}^-)$ and $e(v, L^+) = e(v, \tilde{S}^+) + e(v, \tilde{L}^+)$, for each vertex $v \in H$. Similar to (II) and (III), we can estimate $\sum_{v \in H} \max\{e(v, L^-), e(v, L^+)\}$ with an additive error of $\epsilon^2 \cdot \sum_{v \in H} e(v, \tilde{L}) = \epsilon^2 \cdot e(H, \tilde{L})$.

Let $\widetilde{\mathrm{ALG}}_1$ denote our estimator for ALG_1. Then we have $|\widetilde{\mathrm{ALG}}_1 - \mathrm{ALG}_1| \leq \epsilon^2 \cdot (e(G[\tilde{L}]) + e(\tilde{S}, \tilde{L}) + e(H, \tilde{L})) \leq \epsilon^2 \cdot (e(G[L]) + e(H, L)) \leq \epsilon^2 m$.

If $\mathrm{ALG}_2 = e(H, L) \geq (\frac{1}{2} + \epsilon^2) \cdot \mathrm{OPT}$, then we are done. Hence, without loss of generality, we assume that $\mathrm{ALG}_2 = e(H, L) < (\frac{1}{2} + \epsilon^2) \cdot \mathrm{OPT}$. By the proof of Theorem 6, we have $\mathrm{ALG}_1 \geq (\frac{1}{2} + \frac{\epsilon^2}{16}) \cdot \mathrm{OPT} \geq (\frac{1}{2} + \frac{\epsilon^2}{16}) \cdot \frac{m}{2} \geq \epsilon m$ if we set $\epsilon \leq \frac{1}{4}$. Therefore, $|\widetilde{\mathrm{ALG}}_1 - \mathrm{ALG}_1| \leq \epsilon^2 m \leq \epsilon \cdot \mathrm{ALG}_1$. Since $\mathrm{ALG}_1 \geq (\frac{1}{2} + \frac{\epsilon^2}{16}) \cdot \mathrm{OPT}$ and we can use $\widetilde{\mathrm{ALG}}_1$ to approximate ALG_1 within a multiplicative factor of $(1 \pm \epsilon)$, we are done.

Query Complexity. In the above analysis, the total query complexity for the ϵ-accurate oracle $\mathcal{O}$ is $\mathrm{poly}(1/\epsilon, 1/\delta, \log(1/\delta))$, which is independent of n.