Online Row Sampling from Random Streams

Masataka Gohda[1,a)] Naonori Kakimura[2,b)]

Abstract: This paper studies spectral approximation for a positive semidefinite matrix in the online setting. It is known in [Cohen *et al*. APPROX 2016] that we can construct a spectral approximation of a given $n \times d$ matrix in the online setting if an additive error is allowed. In this paper, we propose an online algorithm that avoids an additive error with the same time and space complexities as the algorithm of Cohen *et al.*, and provides a better upper bound on the approximation size when a given matrix has small rank. In addition, we consider the online random order setting where a row of a given matrix arrives uniformly at random. In this setting, we propose time and space efficient algorithms to find a spectral approximation. Moreover, we reveal that a lower bound on the approximation size in the online random order setting is $\Omega(d\epsilon^{-2}\log n)$, which is larger than the one in the offline setting by an $\mathrm{O}(\log n)$ factor.

Keywords: algorithmic spectral graph theory, spectral approximation, online algorithm, random streams

1. Introduction

Spectral sparsification is to compress the Laplacian matrix of a dense graph to the one of a sparse graph maintaining its quadratic form for an arbitrary vector. It was introduced by Spielman and Teng [38] as a generalization of cut sparsification, and they presented a nearly-linear-time algorithm for spectral sparsification. Since then algorithms for spectral sparsification have become faster and more refined [4], [30], [39] and brought a new paradigm to numerical linear algebra and spectral graph theory. In particular, it led to efficient algorithms for several problems such as linear systems in symmetric diagonally dominant matrices [12], [24], maximum *s-t* flow problems [22], [34], and linear programming [28]. Spectral sparsification has been generalized to positive semidefinite (PSD) matrices [13], [31], [32], which we call *spectral approximation* to distinguish from spectral sparsification. For a matrix $\mathbf{A}$ in $\mathbb{R}^{n\times d}$, a matrix $\widetilde{\mathbf{A}} \in \mathbb{R}^{n'\times d}$ is called a $(1\pm\epsilon)$-*spectral approximation for* $\mathbf{A}$ if, for every $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}^\top\widetilde{\mathbf{A}}^\top\widetilde{\mathbf{A}}\mathbf{x}$ approximates $\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x}$ within a factor of $1 \pm \epsilon$. The number of rows in $\widetilde{\mathbf{A}}$, n', is called the *approximation size*. It is known that there exists an algorithm that returns a $(1 \pm \epsilon)$-spectral approximation with approximation size $\mathrm{O}(d/\epsilon^2)$ [4], [30], while the approximation size is $\Omega(d/\epsilon^2)$ in the worst case [4].

Due to space limitations, we omit almost all proofs and refer the reader to the full version of the paper [17].

1.1 Our Results

Recently, spectral approximation has been studied in restrictive settings such as the semi-streaming setting and the online setting [14], [25]. In the line of research, this paper focuses on the online setting. In the online setting, a matrix $\mathbf{A}$ is not known in advance, and each row of $\mathbf{A}$ arrives one-by-one. Each time we receive a row of $\mathbf{A}$, we decide irrevocably whether to choose the row for a resulting spectral approximation or not and cannot discard or reweight it later. Cohen *et al.* [14] proposed a simple algorithm for an (ϵ,δ)-spectral approximation, where an (ϵ,δ)-*spectral approximation* is a matrix $\widetilde{\mathbf{A}}$ such that $(1-\epsilon)\mathbf{A}^\top\mathbf{A} - \delta\mathbf{I} \preceq \widetilde{\mathbf{A}}^\top\widetilde{\mathbf{A}} \preceq (1+\epsilon)\mathbf{A}^\top\mathbf{A} + \delta\mathbf{I}$. The approximation size is shown to be $\mathrm{O}\left(d\epsilon^{-2}\log d\log\left(\epsilon\|\mathbf{A}\|_2^2/\delta\right)\right)$. They further improved the approximation size to $\mathrm{O}\left(d\epsilon^{-2}\log\left(\epsilon\|\mathbf{A}\|_2^2/\delta\right)\right)$ by an $\mathrm{O}(\log d)$ factor with the aid of a method to obtain linear-sized approximation [30]. This is asymptotically optimal in the sense that no algorithm based on row sampling can obtain an (ϵ,δ)-spectral approximation with $\mathrm{o}\left(d\epsilon^{-2}\log\left(\epsilon\|\mathbf{A}\|_2^2/\delta\right)\right)$ rows [14].

The main contributions of this paper are threefold. First, we revisit the online spectral approximation algorithm by Cohen *et al.* [14], and remove the additional parameter δ to obtain a $(1\pm\epsilon)$-spectral approximation. Second, we consider the case when each row arrives uniformly at random for the online spectral approximation problem, and propose a fast and memory-efficient algorithm that achieves a $(1 \pm \epsilon)$-spectral approximation. Finally, we reveal a lower bound on the approximation size in the online random order setting. Let us describe our results in more detail.

1.1.1 Online setting.

The online spectral approximation algorithm by Cohen *et al.* [14] is simple and optimal with respect to the approximation size, but it entails the additive error δ. It is not difficult to modify their algorithm to the one for finding a $(1 \pm \epsilon)$-spectral approximation by setting $\delta = \epsilon\min_i\left(\sigma_{\min}(\mathbf{A}_i)^2\right)$, where $\mathbf{A}_i$ is the matrix composed of the first i rows in $\mathbf{A}$ and $\sigma_{\min}(\mathbf{A}_i)$ is the smallest non-zero singular value of $\mathbf{A}_i$. However, this requires us to know some estimation of $\min_i\left(\sigma_{\min}(\mathbf{A}_i)^2\right)$ beforehand. Our first result is to present spectral approximation algorithms without such prior information. We propose two algorithms (Theorem 1.1 and Theorem 1.2) by analogy with Cohen *et al.* [14].

[1] The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[2] Keio University, Kohoku, Kanagawa 223–8522, Japan
[a)] masataka_goda@mist.i.u-tokyo.ac.jp
[b)] kakimura@math.keio.ac.jp

To state our results, we denote

$$\mu(\mathbf{A}) \stackrel{\text{def}}{=} \frac{\|\mathbf{A}\|_2^2}{\min_{1\le i\le n}(\sigma_{\min}(\mathbf{A}_i)^2)}. \quad (1)$$

Note that $\mu(\mathbf{A})$ may be 1, e.g., when $\mathbf{A}$ is the identity matrix. We say that an event with $\mathbf{A} \in \mathbb{R}^{n\times d}$ happens *with high probability* if it happens with probability at least $1 - 1/\mathrm{poly}(d)$

Theorem 1.1. *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be a matrix of rank r, and $\epsilon \in (0, 1/2]$ be an error parameter. Then, in the online setting, we can construct with high probability a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$ whose approximation size is $\mathrm{O}\left((r\log\mu(\mathbf{A}) + r + \log d)\,\epsilon^{-2}\log d\right)$.*

Theorem 1.2. *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be a matrix of rank r, and $\epsilon \in (0, 1)$ be an error parameter. Then, in the online setting, we can construct a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$ whose approximation size is $\mathrm{O}\left(r\epsilon^{-2}\log\mu(\mathbf{A}) + r\epsilon^{-2}\right)$ in expectation.*

We remark that, using the same instance as in Theorem 5.1 of Cohen *et al.* [14], it turns out that the approximation size in Theorem 1.2 is asymptotically optimal. That is, no algorithm can obtain a $(1\pm\epsilon)$-spectral approximation with $\mathrm{o}\left(r\epsilon^{-2}\log\mu(\mathbf{A}) + r\epsilon^{-2}\right)$ rows.

Let us compare our algorithms with (ϵ, δ)-spectral approximation algorithms in [14]. If we set $\delta = \epsilon\min_i\sigma_{\min}(\mathbf{A}_i)^2$, then their online algorithms return a $(1 \pm \epsilon)$-spectral approximation whose sizes are $\mathrm{O}\left(d\epsilon^{-2}\log d\log\mu(\mathbf{A})\right)$ and $\mathrm{O}\left(d\epsilon^{-2}\log\mu(\mathbf{A})\right)$, respectively. Therefore, Theorem 1.1 and Theorem 1.2 give better upper bounds on the approximation size as $r \ll d$. Note that the approximation size of [14] always depends on d due to the regularizing factor δ. Note also that the running time and the space complexity are the same.

The framework of our first algorithm is similar to Cohen *et al.* [14]: Each time we receive a row, we compute a score of the arriving row with a matrix we have at the moment, that measures the importance of the arriving row. Then we decide whether to sample the row or not based on the score. Cohen *et al.* [14] used the *online ridge leverage score*, assuming that the current matrix (together with $\delta\mathbf{I}$) is nonsingular. On the other hand, we introduce a new score called a *relative leverage score*, that handles a singular current matrix directly. To analyze relative leverage scores for singular matrices, we exploit the pseudo-determinant [18], [23], which is of independent interest.

Cohen *et al.* [14] improved the approximation size by an $\mathrm{O}(\log d)$ factor, based on a technique to obtain linear-sized approximation introduced in [29]. The linear-sized approximation technique was originally developed in [4], and later made randomized to obtain a faster algorithm [30]. We combine the analysis in Theorem 1.1 with the randomized one to obtain Theorem 1.2.

1.1.2 Online random order setting.

The approximation sizes in Theorem 1.1 and Theorem 1.2 depend on $\mu(\mathbf{A})$. Thus when $\mu(\mathbf{A})$ exceeds $\mathrm{poly}(n)$, the approximation size can be $\omega(r\,\mathrm{poly}(\log n, \epsilon))$. In fact, there exists a matrix $\mathbf{A}$ such that we have to sample all the rows in $\mathbf{A}$ to construct a $(1 \pm \epsilon)$-spectral approximation in the online setting. Moreover, in our algorithms as well as algorithms in [14], we need to compute a Moore-Penrose pseudo-inverse to evaluate the score in each iteration, and thus the running time is not efficient. In fact, even if we exploit the Sharman-Morrison formula for the Moore-Penrose pseudo-inverses, it takes $\mathrm{O}(nd^2)$ time in total. Our second and third contributions are to study the online random order setting to break these difficulties.

In the online random order setting, each row in an input matrix $\mathbf{A}$ comes in exactly once according to a certain random permutation in addition to the online setting. More formally, we are given a family of row vectors $\mathbf{X}$. Let $\mathcal{A}(\mathbf{X})$ be a discrete uniform distribution whose element is a matrix obtained by permuting vectors in $\mathbf{X}$. We note that $\mathcal{A}(\mathbf{X})$ has $n!$ elements. Then the *online random order setting* means that an input matrix is a random variable $\mathbf{A} \sim \mathcal{A}(\mathbf{X})$, and is given as a stream of rows in the online setting. Algorithms in the random order setting have been analyzed for several problems such as frequency moment estimation [6], computation of the median [8], and app roximation of maximum matching in a graph [16], and it is often shown that randomness breaks the worst-case complexity of the online setting.

In the online random order setting, we propose a fast and memory-efficient algorithm that returns a $(1\pm\epsilon)$-spectral approximation. The approximation size is $\mathrm{O}\left(d\epsilon^{-2}\log n\log d\right)$ for almost all rows' permutations, which is independent of $\mu(\mathbf{A})$. We here denote the number of nonzero entries of a matrix $\mathbf{A}$ by $\mathrm{nnz}(\mathbf{A})$.

Theorem 1.3. *Let $\epsilon \in (0, 1/2]$ be an error parameter, and $\mathbf{X}$ be a family of n row vectors in $\mathbb{R}^d$. Then there exists an algorithm in the online random order setting such that $\mathbf{A} \sim \mathcal{A}(\mathbf{X})$ satisfies the following with high probability: The algorithm returns a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$ with $\mathrm{O}\left(d\epsilon^{-2}\log n\log d\right)$ rows with high probability. It consumes $\mathrm{O}\left(\mathrm{nnz}(\mathbf{A})\log n + \left(d^\omega + d^2\log n\right)\log^2 n\ \log d\right)$ time and stores $\mathrm{O}(d\log n\log d)$ rows as the working memory and $\mathrm{O}\left(d\epsilon^{-2}\log n\log d\right)$ rows as the output memory.*

We note that there are two kinds of randomness: random permutation in an input and random sampling in algorithms.

For the proof of Theorem 1.3, we first present a simpler algorithm with less efficient time and space complexity. The idea of our algorithm is simple. Recall that our algorithm in Theorem 1.1 computes the relative leverage score with a current matrix, and samples an arriving row based on that score. The time-consuming part is to compute a pseudo-inverse to obtain the relative leverage score each time a current matrix is updated. In the proposed algorithm, we keep using the same matrix to compute the relative leverage scores for consecutive rows in a batch, which reduces the number of computing a pseudo-inverse. The correctness of the algorithm consists of three parts: (i) the output is a $(1 \pm \epsilon)$-spectral approximation, (ii) the approximation size is bounded, and (iii) the algorithm runs in desired time and space. The first part (i) can be shown with a matrix martingale. The proof is similar to Cohen *et al.* [14], but we need careful analysis due to the fact that we use the relative leverage score with a matrix (which depends on results of previous samples). Note that (i) holds independently of row permutations, that is, (i) holds for any $\mathbf{A} \sim \mathcal{A}(\mathbf{X})$. The randomness of row permutations is exploited to prove (ii) and (iii). We make use of the result in [13] that a matrix obtained by sampling rows uniformly at random from an input

matrix gives a good approximation of leverage scores. Owing to this fact, we prove that the number of computing the Moore-Penrose pseudo-inverses used for the relative leverage scores is reduced to $O(\log n)$, keeping the approximation size small. This simple algorithm, with further observations to reduce time complexity, yields Theorem 1.3. Moreover, we also prove that the simple algorithm, together with a semi-streaming algorithm [25], can reduce the working memory space to $O(d \log d)$ rows, which does not depend on ϵ and n.

We remark that the approximation size can be improved to $O\left(d\epsilon^{-2} \log n\right)$ using the algorithm in Theorem 1.2, although the running time and the space complexity are much less efficient.

1.1.3 Lower bound in the online random order setting.

On the other hand, we obtain a lower bound for the online random order setting. We prove that any algorithm that selects rows in the online random order setting and returns a $(1 \pm \epsilon)$-spectral approximation must sample $\Omega\left(d\epsilon^{-2} \log n\right)$ rows with high probability.Since the lower bound on the approximation size in the offline setting is $\Omega(d\epsilon^{-2})$, the online random order setting suffers an additional $\log n$ factor.

Theorem 1.4. *Let $\epsilon \in (0, 1)$ be an error parameter. Let R be an algorithm that samples rows in the online random order setting and returns a $(1 \pm \epsilon)$-spectral approximation. Then there exists a family of row vectors $\mathbf{X}$ in $\mathbb{R}^d$ such that R with an input $\mathbf{A} \sim \mathcal{A}(\mathbf{X})$ returns $\Omega\left(d\epsilon^{-2} \log n\right)$ rows with high probability.*

We define the worst instance to be the incidence matrix of a graph on d vertices such that every pair of vertices has $n/\binom{d}{2}$ parallel edges. Then the key ingredient is that there exists an integer D such that, if we sample D rows uniformly at random from the instance, then the corresponding matrix is a $(1 \pm \epsilon)$-spectral approximation for a weighted complete graph with high probability. Since a weighted complete graph on d vertices requires $\Omega(d\epsilon^{-2})$ rows for a $(1 \pm \epsilon)$-spectral approximation, this implies that we need to sample $\Omega(d\epsilon^{-2})$ rows while D rows arrive. By dividing the rows of $\mathbf{A}$ into $O(\log n)$ parts with a geometric series $D, 2D, 4D, \ldots$, this yields Theorem 1.4.

1.2 Related work.

The difficulty of spectral approximation in restrictive settings such as the semi-streaming setting and the online setting lies in that the probability of sampling a row becomes dependent on which rows are sampled so far. In the standard spectral approximation (without any restriction), we are given all the rows of a matrix $\mathbf{A}$ in advance. Then we can sample each row by setting a sampling probability with $\mathbf{A}$. The matrix Chernoff bound provides an exponential tail bound, which ensures that the output $\widetilde{\mathbf{A}}$ is a $(1 \pm \epsilon)$-spectral approximation. However, in the semi-streaming or the online setting, the probability that an algorithm returns a spectral approximation is no longer bounded by the matrix Chernoff bound due to the dependencies of the sampled rows. We need to analyze carefully how the output $\widetilde{\mathbf{A}}$ is constructed in the process. Cohen *et al.* [14] and Kyng *et al.* [25] made use of a matrix martingale and its exponential tail bound, in which we are allowed to have mutually dependent random variables. The analysis with a matrix martingale is also used in other papers such as [10], [26], [27]. We also employ it in our settings.

In the dynamic setting, we aim to maintain a spectral approximation under row insertions and deletions. For a special case where $\mathbf{A}$ is the Laplacian matrix, Abraham *et al.* [1] showed that we can construct a $(1 \pm \epsilon)$-spectral approximation such that the amortized update time per insertion or deletion is $O(\mathrm{poly}(\log d, \epsilon))$, which was later de-amortized by [5]. Their algorithms maintain spanners in a graph, which is a different approach from our algorithms based on sampling rows in the online setting. When $\mathbf{A}$ is the Laplacian matrix, we can also obtain a spectral approximation in the dynamic semi-streaming setting [20], [21].

There exist other research directions such as a spectral sketch [2], [19] and spectral sparsification for a generalization of undirected graphs [3], [10], [11], [36]. For a matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$, a *spectral sketch* is a function f such that $(1 - \epsilon)\mathbf{x}^\top \mathbf{A}\mathbf{x} \leq f(\mathbf{x}) \leq (1 + \epsilon)\mathbf{x}^\top \mathbf{A}\mathbf{x}$ for every vector $\mathbf{x}$. If $\mathbf{A}$ is the Laplacian matrix, it is known that there is a nearly-linear time algorithm which returns a spectral sketch with $O(d/\epsilon)$ bits, which is better than $\Omega\left(d/\epsilon^2\right)$ bits of a $(1 \pm \epsilon)$-spectral sparsifier [19]. Recently, spectral sparsification has been extended to directed graphs [10], [11] and hypergraphs [3], [36]. As a practical application, a graph-based learning such as laplacian smoothing and spectral clustering can be made faster by replacing the laplacian matrix with its spectral sparsifier [7].

2. Preliminaries

2.1 Spectral Approximation

We define the $(1 \pm \epsilon)$-*spectral approximation*. Recall that, for two symmetric matrices $\mathbf{A}$, $\mathbf{B}$, we denote $\mathbf{A} \preceq \mathbf{B}$ if $\mathbf{B} - \mathbf{A}$ is a positive semidefinite (PSD) matrix.

Definition 2.1 (Spectral Approximation)**.** *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ be a matrix and $\epsilon \in (0, 1)$ be an error parameter. We say that $\widetilde{\mathbf{A}} \in \mathbb{R}^{m \times d}$ is a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$ if*

$$(1 - \epsilon)\mathbf{A}^\top \mathbf{A} \preceq \widetilde{\mathbf{A}}^\top \widetilde{\mathbf{A}} \preceq (1 + \epsilon)\mathbf{A}^\top \mathbf{A}.$$

Notice that, by the Courant-Fischer theorem, each eigenvalue of $\widetilde{\mathbf{A}}^\top \widetilde{\mathbf{A}}$ approximates the corresponding one of $\mathbf{A}^\top \mathbf{A}$ within a factor of $1 \pm \epsilon$.

For an edge-weighted graph G with d vertices and n edges, we denote the incidence matrix for G by $\mathbf{B}_G \in \mathbb{R}^{n \times d}$, and the Laplacian matrix of G by $\mathbf{L}_G \in \mathbb{R}^{d \times d}$. Thus $\mathbf{L}_G = \mathbf{B}_G^\top \mathbf{B}_G$ holds. A $(1 \pm \epsilon)$-spectral approximation for $\mathbf{B}_G$ is called a $(1 \pm \epsilon)$-*spectral sparsifier of G* .

In spectral approximation algorithms, the *leverage score* plays a major role, which is defined as below. For a matrix $\mathbf{A}$, the i-th row is denoted by $\mathbf{a}_i^\top$, and the Moore-Penrose pseudo-inverse of $\mathbf{A}$ is denoted by $\mathbf{A}^+$.

Definition 2.2 (Leverage Score)**.** *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ be a matrix. For $i \in \{1, 2, \ldots, n\}$, the* leverage score $\tau_i(\mathbf{A})$ *is defined to be*

$$\tau_i(\mathbf{A}) \stackrel{\text{def}}{=} \mathbf{a}_i^\top \left(\mathbf{A}^\top \mathbf{A}\right)^+ \mathbf{a}_i.$$

When an input matrix is the Laplacian matrix of a graph, the leverage score with respect to an edge $e = (a, b)$ is equivalent to the effective resistance between a and b in the graph (see e.g., [37]). We show properties of the leverage score.

Lemma 2.3. *For a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, we have the following properties:*
(i) $0 \le \tau_i(\mathbf{A}) \le 1$ *for any* $i \in \{1, 2, \ldots, n\}$.
(ii) $\sum_{i=1}^{n} \tau_i(\mathbf{A}) = \mathrm{rank}(\mathbf{A})$.

Proof. **(i)** Since $\mathbf{x}^\top (\mathbf{A}^\top\mathbf{A})^+ \mathbf{x} \le \mathbf{x}^\top \left(\mathbf{a}_i\mathbf{a}_i^\top\right)^+ \mathbf{x}$ holds for all $\mathbf{x} \in \mathrm{Im}\left(\mathbf{a}_i\mathbf{a}_i^\top\right)$, we have

$$\tau_i(\mathbf{A}) \le \mathbf{a}_i^\top \left(\mathbf{a}_i\mathbf{a}_i^\top\right)^+ \mathbf{a}_i = \mathrm{tr}\left(\mathbf{a}_i^\top \left(\mathbf{a}_i\mathbf{a}_i^\top\right)^+ \mathbf{a}_i\right) = \mathrm{tr}\left(\left(\mathbf{a}_i\mathbf{a}_i^\top\right)^+ \mathbf{a}_i\mathbf{a}_i^\top\right) = 1.$$

Also $\tau_i(\mathbf{A}) \ge 0$ as $(\mathbf{A}^\top\mathbf{A})^+ \succeq \mathbf{O}$. Hence **(i)** holds.
(ii) It follows that

$$\sum_{i=1}^{n} \tau_i(\mathbf{A}) = \mathrm{tr}\left(\mathbf{A}\left(\mathbf{A}^\top\mathbf{A}\right)^+ \mathbf{A}^\top\right) = \mathrm{tr}\left(\left(\mathbf{A}^\top\mathbf{A}\right)^+ \mathbf{A}^\top\mathbf{A}\right) = \mathrm{rank}(\mathbf{A}).$$

□

The leverage score indicates how important the corresponding row is. In fact, the following theorem asserts that we can construct a spectral approximation with small approximation size via simple random sampling based on the leverage score. More precisely, if we are given a *leverage score overestimate*, that is, a vector $\mathbf{u} \in \mathbb{R}^n$ such that, for all i, $\tau_i(\mathbf{A}) \le u_i$, then the number of rows in $\left(1 \pm 1/\sqrt{\theta}\right)$-spectral approximation is bounded by $O(\theta\|\mathbf{u}\|_1 \log d)$.

Theorem 2.4 (Spectral Approximation via Row Sampling [13])**.** *Let θ be an error parameter, and c be a fixed positive constant. Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be a matrix, and $\mathbf{u} \in \mathbb{R}^n$ be a leverage score overestimate. We define a sampling probability $p_i = \min(\theta \cdot u_i c \log d, 1)$ $(i = 1, \ldots, n)$, and construct a random diagonal matrix $\mathbf{S}$ whose i-th diagonal element is $1/\sqrt{p_i}$ with probability p_i and 0 otherwise. Then, with probability at least $1 - d^{-c/3}$, $\mathbf{SA}$ is a $\left(1 \pm 1/\sqrt{\theta}\right)$-spectral approximation for $\mathbf{A}$ such that $\mathbf{SA}$ contains at most $\sum_i p_i \le \theta\|\mathbf{u}\|_1 c \log d$ non-zero rows.*

For example, suppose that we are given a *constant leverage score overestimate*, that is, a vector $\mathbf{u} \in \mathbb{R}^n$ such that, for all i, $\tau_i(\mathbf{A}) \le u_i \le \beta\tau_i(\mathbf{A})$ for some constant β. Then the above theorem implies that, by setting $\theta = \epsilon^{-2}$, we can obtain a $(1 \pm \epsilon)$-spectral approximation such that it has $O\left(r\epsilon^{-2} \log d\right)$ rows, where $r = \mathrm{rank}(\mathbf{A})$, as $\sum_i u_i \le \beta \sum_i \tau_i(\mathbf{A}) \le \beta r$ by Lemma 2.3.

2.2 Spectral Approximation in the Online Setting

In the online setting, a matrix $\mathbf{A}$ is given as a stream of rows, and we receive a row sequentially. Each time the i-th row $\mathbf{a}_i$ arrives, we irrevocably decide whether $\mathbf{a}_i$ is sampled or not. We cannot access the whole matrix $\mathbf{A}$ in each decision, and thus we cannot compute a standard leverage score. We introduce a variant of the leverage score, called the *relative leverage score*, that can be computed with the matrix we have at the moment (corresponding to a matrix $\mathbf{B}$ in the definition below). This gives a leverage score overestimate.

Definition 2.5 (Relative Leverage Score)**.** *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times d}$ be matrices. For $i \in \{1, 2, \ldots, n\}$, the* relative leverage score $\tau_i^{\mathbf{B}}(\mathbf{A})$ *is defined as follows:*

$$\tau_i^{\mathbf{B}}(\mathbf{A}) \stackrel{\mathrm{def}}{=} \mathbf{a}_i^\top \left(\begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}^\top \begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}\right)^+ \mathbf{a}_i.$$

We note that, if $\mathbf{B}$ is a submatrix consisting of rows of $\mathbf{A}$, then $\tau_i^{\mathbf{B}}(\mathbf{A}) \ge \tau_i(\mathbf{A})$ holds.

The relative leverage score can be rewritten as follows. For vectors $\mathbf{x}$ and $\mathbf{y}$, we denote $\mathbf{x} \perp \mathbf{y}$ if $\mathbf{x}^\top\mathbf{y} = 0$. For a vector $\mathbf{x}$ and a linear subspace W, $\mathbf{x} \perp W$ means that $\mathbf{x} \perp \mathbf{y}$ for all $\mathbf{y} \in W$.

Lemma 2.6. *For $i \in \{1, 2, \ldots, n\}$, it holds that*

$$\tau_i^{\mathbf{B}}(\mathbf{A}) = \begin{cases} \frac{\mathbf{a}_i^\top (\mathbf{B}^\top\mathbf{B})^+ \mathbf{a}_i}{\mathbf{a}_i^\top (\mathbf{B}^\top\mathbf{B})^+ \mathbf{a}_i + 1} & \text{if } \mathbf{a}_i \perp \mathrm{Ker}(\mathbf{B}), \\ 1 & \text{otherwise.} \end{cases}$$

To prove Lemma 2.6, we consider the perturbation of Moore-Penrose pseudo-inverses under the rank-1 update operation given in [33].

Proposition 2.7 (Sherman-Morrison Formula for Moore-Penrose pseudo-inverse [33])**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a* PSD *matrix, $\mathbf{u} \in \mathbb{R}^n$ be a vector, and k be a real-valued multiplier. If $\mathbf{u} \perp \mathrm{Ker}(\mathbf{A})$, then we have*

$$\left(\mathbf{A} + k\mathbf{u}\mathbf{u}^\top\right)^+ = \mathbf{A}^+ - \frac{k\mathbf{A}^+\mathbf{u}\mathbf{u}^\top\mathbf{A}^+}{1 + k\mathbf{u}^\top\mathbf{A}^+\mathbf{u}}.$$

Proof of Lemma 2.6. If $\mathbf{a}_i \perp \mathrm{Ker}(\mathbf{B})$, by Proposition 2.7, we obtain

$$\begin{aligned}\tau_i^{\mathbf{B}}(\mathbf{A}) &= \mathbf{a}_i^\top \left(\begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}^\top \begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}\right)^+ \mathbf{a}_i \\ &= \mathbf{a}_i^\top \left(\left(\mathbf{B}^\top\mathbf{B}\right)^+ - \frac{\left(\mathbf{B}^\top\mathbf{B}\right)^+ \mathbf{a}_i\mathbf{a}_i^\top \left(\mathbf{B}^\top\mathbf{B}\right)^+}{1 + \mathbf{a}_i^\top \left(\mathbf{B}^\top\mathbf{B}\right)^+ \mathbf{a}_i}\right)\mathbf{a}_i \\ &= \frac{\mathbf{a}_i^\top \left(\mathbf{B}^\top\mathbf{B}\right)^+ \mathbf{a}_i}{\mathbf{a}_i^\top \left(\mathbf{B}^\top\mathbf{B}\right)^+ \mathbf{a}_i + 1}. \end{aligned} \tag{2}$$

Next suppose that $\mathbf{a}_i \not\perp \mathrm{Ker}(\mathbf{B})$. Then $\dim\left(\mathrm{Im}\left(\mathbf{B}^\top\mathbf{B} + \mathbf{a}_i\mathbf{a}_i^\top\right)\right)$ is exactly one larger than $\dim\left(\mathrm{Im}\left(\mathbf{B}^\top\mathbf{B}\right)\right)$. Hence there exists a nonzero vector $\mathbf{u} \in \mathrm{Im}\left(\mathbf{B}^\top\mathbf{B} + \mathbf{a}_i\mathbf{a}_i^\top\right)$ such that $\mathbf{u}$ belongs to the orthogonal complement of $\mathrm{Im}\left(\mathbf{B}^\top\mathbf{B}\right)$. By the definition of the pseudo-inverse, we have

$$\mathbf{u} = \left(\begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}^\top \begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}\right)^+ \left(\begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}^\top \begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}\right)\mathbf{u} = \left(\begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}^\top \begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}\right)^+ \mathbf{a}_i\mathbf{a}_i^\top\mathbf{u}.$$

Multiplying $\mathbf{a}_i^\top$ from the left side, we have

$$\mathbf{a}_i^\top\mathbf{u} = \mathbf{a}_i^\top \left(\begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}^\top \begin{pmatrix}\mathbf{B}\\ \mathbf{a}_i^\top\end{pmatrix}\right)^+ \mathbf{a}_i\mathbf{a}_i^\top\mathbf{u} = \tau_i^{\mathbf{B}}(\mathbf{A})\mathbf{a}_i^\top\mathbf{u}.$$

As $\mathbf{a}_i^\top\mathbf{u} \ne 0$, we obtain $\tau_i^{\mathbf{B}}(\mathbf{A}) = 1$. □

From Lemma 2.6, $\tau_i^{\mathbf{B}}(\mathbf{A})$ can be computed with matrix $(\mathbf{B}^\top\mathbf{B})^+$ and $\mathbf{a}_i$. Furthermore, we have $0 \le \tau_i^{\mathbf{B}}(\mathbf{A}) \le 1$, and $\tau_i^{\mathbf{B}}(\mathbf{A})$ is equal to 1 if and only if $\mathbf{a}_i \not\perp \mathrm{Ker}(\mathbf{B})$.

3. $(1 \pm \epsilon)$-spectral approximation algorithm in the online setting

Cohen *et al.* [14] presented an algorithm for an (ϵ, δ)-spectral approximation based on sampling with online ridge leverage scores. In this section, we describe an algorithm that returns a $(1 \pm \epsilon)$-spectral approximation for a given PSD matrix $\mathbf{A}$ as Algorithm 1. Our algorithm gives a better upper bound on the approximation size.

In Algorithm 1, $\widetilde{\mathbf{A}}_i$ is a matrix we have sampled until the end

of the i-th iteration. In the i-th iteration, we determine a sampling probability p_i with the relative leverage score $\tau_i^{\widetilde{\mathbf{A}}_{i-1}}(\mathbf{A})$, and append the arriving row $\mathbf{a}_i$ to $\widetilde{\mathbf{A}}_{i-1}$ with probability p_i to obtain $\widetilde{\mathbf{A}}_i$. This step can be computed with only $\widetilde{\mathbf{A}}_{i-1}$ and $\mathbf{a}_i$. In the end, the algorithm returns $\widetilde{\mathbf{A}}_n$.

Algorithm 1 OnlineRowSampling $(\mathbf{A}, \epsilon)$

Input: a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, an error parameter $\epsilon \in (0, 1/2]$.
Output: a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$.
Define $c = 3\epsilon^{-2} \log d$.
$\widetilde{\mathbf{A}}_0 \leftarrow \mathbf{O}$.
for $i = 1, \ldots, n$ **do**
  $\tilde{l}_i \leftarrow \min\left((1+\epsilon)\tau_i^{\widetilde{\mathbf{A}}_{i-1}}(\mathbf{A}), 1\right)$.
  $p_i \leftarrow \min\left(c\tilde{l}_i, 1\right)$.
  $\widetilde{\mathbf{A}}_i \leftarrow \begin{cases} \begin{pmatrix} \widetilde{\mathbf{A}}_{i-1} \\ \mathbf{a}_i^\top / \sqrt{p_i} \end{pmatrix} & \text{with probability } p_i, \\ \widetilde{\mathbf{A}}_{i-1} & \text{otherwise.} \end{cases}$
end for
return $\widetilde{\mathbf{A}}_n$.

It holds that Algorithm 1 satisfies the conditions of Theorem 1.1: Algorithm 1 returns a $(1\pm\epsilon)$-spectral approximation, and its approximation size is $\mathrm{O}\left((r \log \mu(\mathbf{A}) + r + \log d)\, \epsilon^{-2} \log d\right)$.

For the case when an input matrix is the incidence matrix of a graph, the upper bound in Theorem 1.1 can be simplified as follows.

Corollary 3.1. *Let G be a simple, edge-weighted graph whose largest and smallest weights are $w_{\max}$ and $w_{\min}$, respectively. Let $\mathbf{A}$ be its incidence matrix of rank r. Then Algorithm 1 returns, with high probability, a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$ with $\mathrm{O}\left((r \log (w_{\max}/w_{\min}) + r + \log d)\ \epsilon^{-2} \log d\right)$ edges.*

Regarding a lower bound on the approximation size in the online setting, we show that in order to construct a $(1 \pm \epsilon)$-spectral approximation $\Omega\left(r\epsilon^{-2} \log \mu(\mathbf{A}) + r\epsilon^{-2}\right)$ rows have to be sampled in the worst case. This can be shown in the same way as Theorem 5.1 in [14].

Theorem 3.2. *Let $\epsilon \in (0, 1)$ be an error parameter. Let R be an algorithm that samples rows in the online setting and returns a $(1 \pm \epsilon)$-spectral approximation with probability at least 1/2. Then there exists a matrix $\mathbf{A}$ of rank r in $\mathbb{R}^{n\times d}$ such that R samples $\Omega\left(r\epsilon^{-2} \log \mu(\mathbf{A}) + r\epsilon^{-2}\right)$ rows in expectation.*

4. Fast $(1\pm\epsilon)$-approximation in the online random order setting

In Sections 4 and 5, we focus on the online random order setting. Recall that, in the online random order setting, we are given a family of row vectors $\mathbf{X} = \{\mathbf{x}_1^\top, \mathbf{x}_2^\top, \ldots, \mathbf{x}_n^\top\}$ in $\mathbb{R}^d$. An input matrix $\mathbf{A}$ is chosen from a discrete uniform distribution $\mathcal{A}(\mathbf{X})$ whose element is a matrix obtained by permuting row vectors of $\mathbf{X}$, and is given as a stream of rows in the online setting.

In Section 4.1, we present a simpler algorithm (Algorithm 2) such that it returns a $(1 \pm \epsilon)$-spectral approximation with approximation size $\mathrm{O}(d\epsilon^{-2} \log n \log d)$, but it runs in less efficient time and space complexity. In Section 4.2, with further observations, we develop Algorithm 2 into a less running time and space algorithm, which gives Theorem 1.3. Moreover, in Section 4.3, we reduce the space complexity with a semi-streaming algorithm.

4.1 Simple Scaled Sampling Algorithm

We define some notations used in Algorithm 2. Define $K \stackrel{\text{def}}{=} d \log d$. For convenience, we may assume that there exists $\alpha \in \mathbb{N}$ such that $n = (2^{\alpha+1} - 1)K$. Thus $\alpha = \log_2(n/K + 1) - 1$. We also assume that $c_1 \exp(d) > n$ for some fixed positive constant c_1.

In Algorithm 2, we divide $\mathbf{A}$ into blocks. For $i \in \{0, 1, \ldots, \alpha\}$, the *$i$-th block* is a matrix composed of $2^i K$ consecutive rows from $\mathbf{a}_{(2^i-1)K+1}$ to $\mathbf{a}_{(2^{i+1}-1)K}$. Moreover, we denote the matrix consisting of the $0, \ldots, i$-th blocks by $\mathbf{M}_i$. Similarly to Algorithm 1, we denote by $\widetilde{\mathbf{A}}_j$ a matrix we have sampled until the j-th row arrives, and the output is $\widetilde{\mathbf{A}}_n$. Additionally, in the i-th block, we keep a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{M}_{i-1}$, denoted by $\widetilde{\mathbf{M}}_{i-1}$. In the i-th block, we compute $\tau_j^{\widetilde{\mathbf{M}}_{i-1}}(\mathbf{A})$ for a row j, and sample the j-th row based on it. Thus we do not need to compute the Moore-Penrose pseudo-inverse each time, but need the Moore-Penrose pseudo-inverse of only one matrix $\widetilde{\mathbf{M}}_{i-1}^\top \widetilde{\mathbf{M}}_{i-1}$ for the i-th block.

Algorithm 2 ScaledSampling $(\mathbf{A}, \epsilon)$

Input: a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, an error parameter $\epsilon \in (0, 1/2]$.
Output: a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$.
Define $K = d \log d$, $c = 6\epsilon^{-2} \log d$ and $\alpha = \log_2(n/K + 1) - 1$.
$\widetilde{\mathbf{A}}_0 \leftarrow \mathbf{O}$.
for $j = 1, \ldots, K$ **do**
  $\widetilde{\mathbf{A}}_j \leftarrow \begin{pmatrix} \widetilde{\mathbf{A}}_{j-1} \\ \mathbf{a}_j^\top \end{pmatrix}$.
end for
for $i = 1, \ldots, \alpha$ **do**
  $\widetilde{\mathbf{M}}_{i-1} \leftarrow \widetilde{\mathbf{A}}_{(2^i-1)K}$.
  for $j = \left(2^i - 1\right)K + 1, \ldots, \left(2^{i+1} - 1\right)K$ **do**
    $\tilde{l}_j \leftarrow \min\left((1+\epsilon)\tau_j^{\widetilde{\mathbf{M}}_{i-1}}(\mathbf{A}), 1\right)$.
    $p_j \leftarrow \min\left(c\tilde{l}_j, 1\right)$.
    $\widetilde{\mathbf{A}}_j \leftarrow \begin{cases} \begin{pmatrix} \widetilde{\mathbf{A}}_{j-1} \\ \mathbf{a}_j^\top / \sqrt{p_j} \end{pmatrix} & \text{with probability } p_j, \\ \widetilde{\mathbf{A}}_{j-1} & \text{otherwise.} \end{cases}$
  end for
end for
return $\widetilde{\mathbf{A}}_n$.

Algorithm 2 satisfies the following.

Theorem 4.1. *Let $\epsilon \in (0, 1/2]$ be an error parameter, and $\mathbf{X}$ be a family of n row vectors in $\mathbb{R}^d$. Then $\mathbf{A} \sim \mathcal{A}(\mathbf{X})$ satisfies the following with high probability: Algorithm 2 returns a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$ with $\mathrm{O}\left(d\epsilon^{-2} \log n \log d\right)$ rows with high probability. It consumes $\mathrm{O}\left(\mathrm{nnz}(\mathbf{A}) \log n + \left(d^\omega + d^2 \log n\right)\epsilon^{-2} \log^2 n \log d\right)$ time and stores $\mathrm{O}\left(d\epsilon^{-2} \log n \log d\right)$ rows.*

4.2 Fast $(1 \pm \epsilon)$-spectral approximation

In Section 4.2 and Section 4.3, we improve Algorithm 2 to obtain a time and space efficient algorithm. In Algorithm 2, a $(1 \pm \epsilon)$-spectral approximation $\widetilde{\mathbf{M}}_{i-1}$ for $\mathbf{M}_{i-1}$ is used for computing a leverage score overestimate for the i-th block. However, in the proof of Theorem 4.1, in order to construct a $(1 \pm \epsilon)$-spectral approximation, a *constant* spectral approximation for $\mathbf{M}_{i-1}$ suffices. Therefore, by computing leverage score overestimates with a constant spectral approximation instead of a $(1 \pm \epsilon)$-spectral approximation, we can reduce its running time and working memory. Let ConstApprox denote a stream that runs a procedure to

maintain a constant approximation. Then the algorithm is described as Algorithm 3.

Algorithm 3 ImprovedScaledSampling $(\mathbf{A}, \epsilon)$

Input: a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, an error parameter $\epsilon \in (0, 1/2]$.
Output: a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$.
Define $K = d\log d$, $c = 6\epsilon^{-2}\log d$, $\alpha = \log_2(n/K + 1) - 1$.
$\widetilde{\mathbf{A}}_0 \leftarrow \mathbf{O}$.
for $j = 1, \ldots, K$ **do**
  $\widetilde{\mathbf{A}}_j \leftarrow \begin{pmatrix} \widetilde{\mathbf{A}}_{j-1} \\ \mathbf{a}_j^\top \end{pmatrix}$.
  ConstApprox.add($\mathbf{a}_j^\top$).
end for
for $i = 1, \ldots, \alpha$ **do**
  $\widetilde{\mathbf{M}}_{i-1}^c \leftarrow$ ConstApprox.query().
  $\tilde{l}_j \leftarrow \min\left(2\tau_j^{\widetilde{\mathbf{M}}_{i-1}^c}(\mathbf{A}), 1\right)$.
  $p_j \leftarrow \min\left(c\tilde{l}_j, 1\right)$.
  $\widetilde{\mathbf{A}}_j \leftarrow \begin{cases} \begin{pmatrix} \widetilde{\mathbf{A}}_{j-1} \\ \mathbf{a}_j^\top/\sqrt{p_j} \end{pmatrix} & \text{with probability } p_j, \\ \widetilde{\mathbf{A}}_{j-1} & \text{otherwise.} \end{cases}$
  ConstApprox.add($\mathbf{a}_j^\top$).
end for
return $\widetilde{\mathbf{A}}_n$.

In Algorithm 3, ConstApprox.add($\mathbf{a}_j^\top$) means that we add $\mathbf{a}_j^\top$ to the stream ConstApprox. Moreover, ConstApprox.query() returns the current spectral approximation obtained by ConstApprox. In the algorithm, we store $\widetilde{\mathbf{A}}_j$ in the output memory, as $\widetilde{\mathbf{A}}_j$ stores a family of output rows and is never referred. In contrast, $\widetilde{\mathbf{M}}_{i-1}^c$ obtained by ConstApprox is stored in the working memory. As a result, Algorithm 3 satisfies Theorem 1.3 when ConstApprox is set to ScaledSampling $(\mathbf{A}, 1/2)$.

4.3 Memory-efficient $(1 \pm \epsilon)$-spectral approximation

In this section, we achieve a further improvement in the working memory. The required working space is $\mathrm{O}(d\log d)$ rows, which does not depend on ϵ and n. In the Algorithm 3, instead of keeping a constant spectral approximation with ScaledSampling $(\mathbf{A}, 1/2)$, we run a spectral approximation algorithm in the semi-streaming setting in parallel to obtain a $(1 \pm 1/3)$-spectral approximation for $\mathbf{M}_{i-1}$. This is then used to compute a leverage score overestimate, which reduces the working memory space.

It is known as below that there exists an efficient semi-streaming algorithm for spectral approximation [25].

Theorem 4.2 (Sparsification in the Semi-Streaming Setting [25])**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be a matrix, and $\epsilon \in (0, 1/2)$ be an error parameter. In the semi-streaming setting, we can construct, with high probability, a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$ with $\mathrm{O}(d\epsilon^{-2}\log d)$ rows by storing $\mathrm{O}(d\epsilon^{-2}\log d)$ rows in $\mathrm{O}\left(nd^{\omega-1} + nd\log\left(d\epsilon^{-1}\right)\right)$ time.*

Note that the above algorithm maintains a $(1 \pm \epsilon)$-spectral approximation at all times and runs in $\mathrm{O}\left(n\log^2 d\right)$ time if $\mathbf{A}$ is the incidence matrix of a graph.

Let StreamSparsify $(\mathbf{A}, \epsilon)$ denote an algorithm which satisfies the above theorem. We prove that Algorithm 3 satisfies the following theorem when ConstApprox is set to StreamSparsify $(\mathbf{A}, 1/3)$.

Theorem 4.3. *Let $\epsilon \in (0, 1/2]$ be an error parameter, and $\mathbf{X}$ be a family of n row vectors in $\mathbb{R}^d$. Then $\mathbf{A} \sim \mathcal{A}(\mathbf{X})$ satisfies the following with high probability: If we set* ConstApprox *to* StreamSparsify $(\mathbf{A}, 1/3)$*, Algorithm 3 returns a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{A}$ with $\mathrm{O}\left(d\epsilon^{-2}\log n\log d\right)$ rows with high probability. It consumes $\mathrm{O}\left(nd^{\omega-1} + nd\log d + \mathrm{nnz}(\mathbf{A})\log n\right)$ time, where $\mathrm{nnz}(\mathbf{A})$ is the number of non-zero entries in $\mathbf{A}$, and stores $\mathrm{O}(d\log d)$ rows as the working memory and $\mathrm{O}\left(d\epsilon^{-2}\log n\log d\right)$ rows as the output memory.*

Finally, for the case when an input matrix is the incidence matrix of a graph, the running time can be rewritten as follows.

Corollary 4.4. *Let $\epsilon \in (0, 1/2]$ be an error parameter, and $\mathbf{X}$ be a family of n row vectors in $\mathbb{R}^d$ corresponding to the incidence matrix of an undirected graph. If we set* ConstApprox *in Algorithm 3 to* StreamSparsify $(\mathbf{A}, 1/3)$*, Algorithm 3 runs in $\mathrm{O}(n\log n)$ time.*

5. Lower bound in the online random order setting

In this section, we prove Theorem 1.4, that is, in order to maintain a $(1 \pm \epsilon)$-spectral approximation in the online random order setting, any randomized algorithm requires to keep $\Omega\left(d\epsilon^{-2}\log n\right)$ rows in the worst case. Recall that the lower bound on the approximation size without any restriction is $\Omega(d\epsilon^{-2})$ [4], and thus the online random order setting suffers an additional $\log n$ factor. We assume the adversary who knows only how an algorithm works and does not know any result after we run the algorithm. In other words, the worst input stream is determined in advance.

We define an input family of vectors $\mathbf{X}^*$ as follows. Let K_d be a complete graph on d vertices. Then $\mathbf{X}^*$ is defined to be the incidence matrix of $n/\binom{d}{2}K_d$, where, for a graph G and a non-negative number α, αG is a graph obtained from G by making $\alpha - 1$ copies of each edge. That is, $\mathbf{X}^*$ has $n/\binom{d}{2}$ copies of each row in the incidence matrix $\mathbf{B}_{K_d}$. We will prove that the family $\mathbf{X}^*$ gives a lower bound in Theorem 1.4.

We first show in Lemma 5.2 below that there exists a constant D such that $\mathbf{A} \sim \mathcal{A}(\mathbf{X}^*)$ satisfies with high probability that the submatrix $\mathbf{A}_D$, which is the matrix composed of the first D rows in $\mathbf{A}$, is a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{B}_{D/\binom{d}{2}K_d}$. This can be done by regarding sampling in the online random order setting as sampling without replacement from a finite population.

Lemma 5.1 (Tail Bound for the Hypergeometric Distribution [35])**.** *Let C be a set of M elements that contains K 1's and $M-K$ 0's. Let $X_1, \ldots, X_m$ denote the values drawn from C without replacement. Define $S_i \overset{\text{def}}{=} \sum_{j=1}^{i} X_j$ and $\mu \overset{\text{def}}{=} K/M$. Then for all $t > 0$, we have*

$$\mathbb{P}\left(|S_m - m\mu| \geq mt\right) \leq 2\exp\left(-\frac{2mt^2}{1 - f_m^*}\right),$$

where $f_m^* = (m-1)/M$.

Lemma 5.2. *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be a matrix chosen from the uniform distribution $\mathcal{A}(\mathbf{X}^*)$, and $\epsilon \in (0, 1)$ be an error parameter. Set $D = d^4\epsilon^{-2}\log d$ and $\alpha = D/\binom{d}{2}$. Then, with high probability, $\mathbf{A}_D$ is a $(1 \pm \epsilon)$-spectral approximation for $\mathbf{B}_{\alpha K_d}$.*

Proof. Let $S_D(e)$ be the number of an edge e of K_d in $\mathbf{A}_D$. We show $(1-\epsilon)\alpha \leq S_D(e) \leq (1+\epsilon)\alpha$ with high probability, where we

note that α is the expected value of $S_D(e)$. We apply Lemma 5.1, where $M = n$, $m = D$, $\mu = 1/\binom{d}{2}$, and $S_m = S_D(e)$. Setting $t = \epsilon/\binom{d}{2} = \epsilon\alpha/D$, we have

$$\begin{aligned}\mathbb{P}\left(|S_D(e) - \alpha| \geq \epsilon\alpha\right) &\leq 2\exp\left(-\frac{N}{N-D+1}\cdot\frac{2d^4\log d}{\epsilon^2}\left(\frac{\epsilon}{\binom{d}{2}}\right)^2\right)\\ &\leq 2\exp(-8\log d)\\ &= \frac{2}{d^8}.\end{aligned}$$

Thus $(1-\epsilon)\alpha \leq S_D(e) \leq (1+\epsilon)\alpha$ with probability at least $1-2/d^8$. Since K_d has $O(d^2)$ edges, the above inequality holds for all edges in K_d with high probability, by taking a union bound.

Therefore, the graph corresponding to $\mathbf{A}_D$ satisfies that the weight of every edge is between $(1-\epsilon)\alpha$ and $(1+\epsilon)\alpha$ with high probability, which means that $\mathbf{A}_D$ is a $(1\pm\epsilon)$-spectral approximation for $\mathbf{B}_{\alpha K_d}$. □

Let R be an algorithm that returns a $(1\pm\epsilon)$-spectral approximation. Define $s_i = 2^i D$ for $i = 0, 1, \ldots, \lfloor\log_2(n/D)\rfloor$. Since R returns a $(1\pm\epsilon)$-spectral approximation for any instance, R has to keep a $(1\pm\epsilon)$-spectral approximation for $\mathbf{A}_j$ for all $j = 1, \ldots, n$. Hence, for all $i = 0, \ldots, \lfloor\log_2(n/D)\rfloor$, our matrix $\widetilde{\mathbf{A}}_{s_i}$ is a $(1\pm\epsilon)$-spectral approximation for $\mathbf{A}_{s_i}$. On the other hand, it follows from Lemma 5.2 that $\mathbf{A}_{s_i}$ must be a $(1\pm\epsilon)$-spectral approximation for $\mathbf{B}_{\alpha_i K_d}$ for some α_i. They imply that we have to sample $\Omega(d\epsilon^{-2})$ rows between s_i-th and s_{i+1}-th rows.

Proof of Theorem 1.4. Set $\mathbf{X}$ to the family $\mathbf{X}^*$ defined above, and $A \sim \mathcal{A}(\mathbf{X})$. We assume that $\epsilon^2 n > d^4\log d$ and $\epsilon^2 d > c_1$ and $\exp(d^{c_2}) > n$ for fixed positive constants c_1 and c_2. Define $s_i = 2^i D$, $\alpha_i = s_i/\binom{d}{2}$ for $i = 0, 1, \ldots, \lfloor\log_2(n/D)\rfloor$, where $D = d^4\epsilon^{-2}\log d$. We remark that since $\epsilon^2 n > d^4\log d$, it holds that $\lfloor\log_2(n/D)\rfloor \geq 0$.

Let $\widetilde{\mathbf{A}}_j$ be a matrix that an algorithm R maintains after the j-th row arrived. Since R returns a $(1\pm\epsilon)$-spectral approximation for any matrix, $\widetilde{\mathbf{A}}_{s_i}$ is a $(1\pm\epsilon)$-spectral approximation for $\mathbf{A}_{s_i}$ for all $i = 0, 1, \ldots, \lfloor\log_2(n/D)\rfloor$:

$$(1-\epsilon)\mathbf{A}_{s_i}^\top\mathbf{A}_{s_i} \preceq \widetilde{\mathbf{A}}_{s_i}^\top\widetilde{\mathbf{A}}_{s_i} \preceq (1+\epsilon)\mathbf{A}_{s_i}^\top\mathbf{A}_{s_i}. \tag{3}$$

We first suppose that for all $i = 0, 1, \ldots, \lfloor\log_2(n/D)\rfloor$, $\mathbf{A}_{s_i}$ is a $(1\pm\epsilon)$-spectral approximation for $\mathbf{B}_{\alpha_i K_d}$ simultaneously:

$$(1-\epsilon)\mathbf{L}_{\alpha_i K_d} \preceq \mathbf{A}_{s_i}^\top\mathbf{A}_{s_i} \preceq (1+\epsilon)\mathbf{L}_{\alpha_i K_d}. \tag{4}$$

(We evaluate the probability that it holds in the end of the proof.) From Eq. (3) and Eq. (4), we obtain

$$(1-3\epsilon)\mathbf{L}_{\alpha_i K_d} \preceq \widetilde{\mathbf{A}}_{s_i}^\top\widetilde{\mathbf{A}}_{s_i} \preceq (1+3\epsilon)\mathbf{L}_{\alpha_i K_d}. \tag{5}$$

Similarly, we obtain

$$(1-3\epsilon)\mathbf{L}_{\alpha_{i+1} K_d} \preceq \widetilde{\mathbf{A}}_{s_{i+1}}^\top\widetilde{\mathbf{A}}_{s_{i+1}} \preceq (1+3\epsilon)\mathbf{L}_{\alpha_{i+1} K_d}. \tag{6}$$

Subtracting Eq. (5) from Eq. (6) in both sides, we obtain

$$(1-9\epsilon)\mathbf{L}_{\alpha_i K_d} \preceq \widetilde{\mathbf{A}}_{s_{i+1}}^\top\widetilde{\mathbf{A}}_{s_{i+1}} - \widetilde{\mathbf{A}}_{s_i}^\top\widetilde{\mathbf{A}}_{s_i} \preceq (1+9\epsilon)\mathbf{L}_{\alpha_i K_d}.$$

Hence the rows sampled between s_i-th and s_{i+1}-th rows in algorithm R form a $(1\pm 9\epsilon)$-spectral approximation for $\mathbf{B}_{\alpha_i K_d}$. Since it requires $\Omega\left(d\epsilon^{-2}\right)$ rows to construct a $(1\pm O(\epsilon))$-spectral approximation for $\mathbf{B}_{K_d}$ when $\epsilon^2 d > c_1$ [4], the number of rows sampled between s_i-th and s_{i+1}-th rows is $\Omega\left(d\epsilon^{-2}\right)$. Aggregating it for all $i = 0, 1, \ldots, \lfloor\log_2(n/D)\rfloor - 1$, we see that the approximation size is $\Omega\left(d\epsilon^{-2}\log n\right)$.

Finally, we evaluate the probability that for all $i = 0, 1, \ldots, \lfloor\log_2(n/D)\rfloor$, $\mathbf{A}_{s_i}$ is a $(1\pm\epsilon)$-spectral approximation for $\mathbf{B}_{\alpha_i K_d}$. By Lemma 5.2, $\mathbf{A}_{s_i}$ is with high probability a $(1\pm\epsilon)$-spectral approximation for $\mathbf{B}_{\alpha_i K_d}$. Since we assume that $\exp(d^{c_2}) > n$, it holds that $\log_2(n/D) = O(d)$. Taking a union bound, $\mathbf{A}_{s_i}$ is with high probability a $(1\pm\epsilon)$-spectral approximation for $\mathbf{B}_{\alpha_i K_d}$ for all $i = 0, 1, \ldots, \lfloor\log_2(n/D)\rfloor$. In summary, R must sample $\Omega\left(d\epsilon^{-2}\log n\right)$ rows with high probability. □

References

[1] Abraham, I., Durfee, D., Koutis, I., Krinninger, S. and Peng, R.: On Fully Dynamic Graph Sparsifiers, Dinur [15], pp. 335–344 (online), available from ⟨https://ieeexplore.ieee.org/xpl/conhome/7781469/proceeding⟩.

[2] Andoni, A., Chen, J., Krauthgamer, R., Qin, B., Woodruff, D. P. and Zhang, Q.: On Sketching Quadratic Forms, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016* (Sudan, M., ed.), ACM, pp. 311–319 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2840728⟩ (2016).

[3] Bansal, N., Svensson, O. and Trevisan, L.: New Notions and Constructions of Sparsification for Graphs and Hypergraphs, *CoRR*, Vol. abs/1905.01495 (online), available from ⟨http://arxiv.org/abs/1905.01495⟩ (2019).

[4] Batson, J. D., Spielman, D. A. and Srivastava, N.: Twice-ramanujan sparsifiers, *STOC*, ACM, pp. 255–262 (2009).

[5] Bernstein, A., Forster, S. and Henzinger, M.: A Deamortization Approach for Dynamic Spanner and Dynamic Maximal Matching, Chan [9], pp. 1899–1918 (online), DOI: 10.1137/1.9781611975482.

[6] Braverman, V., Viola, E., Woodruff, D. P. and Yang, L. F.: Revisiting Frequency Moment Estimation in Random Order Streams, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic* (Chatzigiannakis, I., Kaklamanis, C., Marx, D. and Sannella, D., eds.), LIPIcs, Vol. 107, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 25:1–25:14 (online), available from ⟨http://www.dagstuhl.de/dagpub/978-3-95977-076-7⟩ (2018).

[7] Carlson, C., Kolla, A., Srivastava, N. and Trevisan, L.: Optimal Lower Bounds for Sketching Graph Cuts, Chan [9], pp. 2565–2569 (online), DOI: 10.1137/1.9781611975482.

[8] Chakrabarti, A., Jayram, T. S. and Patrascu, M.: Tight lower bounds for selection in randomly ordered streams, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008* (Teng, S., ed.), SIAM, pp. 720–729 (online), available from ⟨http://dl.acm.org/citation.cfm?id=1347082⟩ (2008).

[9] Chan, T. M.(ed.): *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, SIAM (2019).

[10] Cohen, M. B., Kelner, J. A., Kyng, R., Peebles, J., Peng, R., Rao, A. B. and Sidford, A.: Solving Directed Laplacian Systems in Nearly-Linear Time through Sparse LU Factorizations, Thorup [40], pp. 898–909 (online), available from ⟨https://ieeexplore.ieee.org/xpl/conhome/8554191/proceeding⟩.

[11] Cohen, M. B., Kelner, J. A., Peebles, J., Peng, R., Rao, A. B., Sidford, A. and Vladu, A.: Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs, *STOC*, ACM, pp. 410–419 (2017).

[12] Cohen, M. B., Kyng, R., Miller, G. L., Pachocki, J. W., Peng, R., Rao, A. B. and Xu, S. C.: Solving SDD linear systems in nearly $m\log^{1/2} n$ time, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014* (Shmoys, D. B., ed.), ACM, pp. 343–352 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2591796⟩ (2014).

[13] Cohen, M. B., Lee, Y. T., Musco, C., Musco, C., Peng, R. and Sidford, A.: Uniform Sampling for Matrix Approximation, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015* (Roughgarden, T., ed.), ACM, pp. 181–190 (online), available from

⟨http://dl.acm.org/citation.cfm?id=2688073⟩ (2015).

[14] Cohen, M. B., Musco, C. and Pachocki, J. W.: Online Row Sampling, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France* (Jansen, K., Mathieu, C., Rolim, J. D. P. and Umans, C., eds.), LIPIcs, Vol. 60, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 7:1–7:18 (online), available from ⟨http://www.dagstuhl.de/dagpub/978-3-95977-018-7⟩ (2016).

[15] Dinur, I.(ed.): *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, IEEE Computer Society (2016).

[16] Farhadi, A., Hajiaghayi, M., Mai, T., Rao, A. and Rossi, R. A.: Approximate Maximum Matching in Random Streams, *http://ryanrossi.com/pubs/soda20.pdf* (2019).

[17] Gohda, M. and Kakimura, N.: Online Spectral Approximation in Random Order Streams, *arXiv preprint arXiv:1911.08800* (2019).

[18] Holbrook, A.: Differentiating the pseudo determinant, *Linear Algebra and its Applications*, Vol. 548, pp. 293–304 (2018).

[19] Jambulapati, A. and Sidford, A.: Efficient $\tilde{O}(n/\epsilon)$ Spectral Sketches for the Laplacian and its Pseudoinverse, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018* (Czumaj, A., ed.), SIAM, pp. 2487–2503 (online), DOI: 10.1137/1.9781611975031 (2018).

[20] Kapralov, M., Lee, Y. T., Musco, C., Musco, C. and Sidford, A.: Single Pass Spectral Sparsification in Dynamic Streams, *FOCS*, IEEE Computer Society, pp. 561–570 (2014).

[21] Kapralov, M., Nouri, N., Sidford, A. and Tardos, J.: Dynamic Streaming Spectral Sparsification in Nearly Linear Time and Space, *CoRR*, Vol. abs/1903.12150 (2019).

[22] Kelner, J. A., Lee, Y. T., Orecchia, L. and Sidford, A.: An Almost-Linear-Time Algorithm for Approximate Max Flow in Undirected Graphs, and its Multicommodity Generalizations, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pp. 217–226 (2014).

[23] Knill, O.: Cauchy–Binet for pseudo-determinants, *Linear Algebra and Its Applications*, Vol. 459, pp. 522–547 (2014).

[24] Koutis, I., Miller, G. L. and Peng, R.: A Nearly-m log n Time Solver for SDD Linear Systems, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011* (Ostrovsky, R., ed.), IEEE Computer Society, pp. 590–598 (online), available from ⟨https://ieeexplore.ieee.org/xpl/conhome/6108120/proceeding⟩ (2011).

[25] Kyng, R., Pachocki, J., Peng, R. and Sachdeva, S.: A Framework for Analyzing Resparsification Algorithms, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19* (Klein, P. N., ed.), SIAM, pp. 2032–2043 (online), DOI: 10.1137/1.9781611974782 (2017).

[26] Kyng, R. and Sachdeva, S.: Approximate Gaussian Elimination for Laplacians - Fast, Sparse, and Simple, Dinur [15], pp. 573–582 (online), available from ⟨https://ieeexplore.ieee.org/xpl/conhome/7781469/proceeding⟩.

[27] Kyng, R. and Song, Z.: A Matrix Chernoff Bound for Strongly Rayleigh Distributions and Spectral Sparsifiers from a few Random Spanning Trees, Thorup [40], pp. 373–384 (online), available from ⟨https://ieeexplore.ieee.org/xpl/conhome/8554191/proceeding⟩.

[28] Lee, Y. T. and Sidford, A.: Path Finding Methods for Linear Programming: Solving Linear Programs in $\tilde{O}$(vrank) Iterations and Faster Algorithms for Maximum Flow, *FOCS*, IEEE Computer Society, pp. 424–433 (2014).

[29] Lee, Y. T. and Sun, H.: Constructing Linear-Sized Spectral Sparsification in Almost-Linear Time, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015* (Guruswami, V., ed.), IEEE Computer Society, pp. 250–269 (online), available from ⟨https://ieeexplore.ieee.org/xpl/conhome/7352273/proceeding⟩ (2015).

[30] Lee, Y. T. and Sun, H.: An SDP-based algorithm for linear-sized spectral sparsification, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017* (Hatami, H., McKenzie, P. and King, V., eds.), ACM, pp. 678–687 (online), DOI: 10.1145/3055399 (2017).

[31] Li, M., Miller, G. L. and Peng, R.: Iterative Row Sampling, *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, IEEE Computer Society, pp. 127–136 (online), available from ⟨https://ieeexplore.ieee.org/xpl/conhome/6685222/proceeding⟩ (2013).

[32] Magdon-Ismail, M.: Row Sampling for Matrix Algorithms via a Non-Commutative Bernstein Bound, *CoRR*, Vol. abs/1008.0587 (2010).

[33] Meyer, Jr, C. D.: Generalized inversion of modified matrices, *SIAM Journal on Applied Mathematics*, Vol. 24, No. 3, pp. 315–323 (1973).

[34] Peng, R.: Approximate Undirected Maximum Flows in $O(m\text{polylog}(n))$ Time, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016* (Krauthgamer, R., ed.), SIAM, pp. 1862–1867 (online), DOI: 10.1137/1.9781611974331 (2016).

[35] Serfling, R. J.: Probability inequalities for the sum in sampling without replacement, *The Annals of Statistics*, pp. 39–48 (1974).

[36] Soma, T. and Yoshida, Y.: Spectral Sparsification of Hypergraphs, *SODA*, SIAM, pp. 2570–2581 (2019).

[37] Spielman, D. A. and Srivastava, N.: Graph Sparsification by Effective Resistances, *SIAM J. Comput.*, Vol. 40, No. 6, pp. 1913–1926 (2011).

[38] Spielman, D. A. and Teng, S.: Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004* (Babai, L., ed.), ACM, pp. 81–90 (2004).

[39] Spielman, D. A. and Teng, S.: Spectral Sparsification of Graphs, *SIAM J. Comput.*, Vol. 40, No. 4, pp. 981–1025 (2011).

[40] Thorup, M.(ed.): *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, IEEE Computer Society (2018).