STREAMING ALGORITHMS FOR ℓ_p FLOWS AND ℓ_p REGRESSION

Amit Chakrabarti
Department of Computer Science
Dartmouth College
Hanover, NH 03755, USA

Jeffrey Jiang
Department of Computer Science
Dartmouth College
Hanover, NH 03755, USA

David P. Woodruff*
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dwoodruf@cs.cmu.edu

Taisuke Yasuda
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
taisukey@cs.cmu.edu

ABSTRACT

We initiate the study of one-pass streaming algorithms for underdetermined ℓ_p linear regression problems of the form

$$\min_{\boldsymbol{Ax}=\boldsymbol{b}} \|\boldsymbol{x}\|_p, \qquad \text{where } \boldsymbol{A} \in \mathbb{R}^{n\times d} \text{ with } n \ll d,$$

which generalizes basis pursuit ($p=1$) and least squares solutions to underdetermined linear systems ($p=2$). We study the column-arrival streaming model, in which the columns of $\boldsymbol{A}$ and then the vector $\boldsymbol{b}$ are presented one by one in a stream. When $\boldsymbol{A}$ is the incidence matrix of a graph, this corresponds to an edge insertion graph stream, and the regression problem captures ℓ_p flows which includes transshipment ($p=1$), electrical flows ($p=2$), and max flow ($p=\infty$) on undirected graphs as special cases. Our goal is to design algorithms which use space much less than the entire stream, which has a length of d. For the task of estimating the cost of the ℓ_p regression problem for $p \in [2,\infty]$, we show a streaming algorithm which constructs a sparse instance supported on $\tilde{O}(\varepsilon^{-2}n)$ columns of $\boldsymbol{A}$ which approximates the cost up to a $(1\pm\varepsilon)$ factor, which corresponds to $\tilde{O}(\varepsilon^{-2}n^2)$ bits of space in general and an $\tilde{O}(\varepsilon^{-2}n)$ space semi-streaming algorithm for constructing ℓ_p flow sparsifiers on graphs. This extends to $p \in (1,2)$ with $\tilde{O}(\varepsilon^2 n^{q/2})$ columns, where q is the Hölder conjugate exponent of p. For $p=2$, we show that $\Omega(n^2)$ bits of space are required in general even for outputting a constant factor solution. For $p=1$, we show that the cost cannot be estimated even to an $o(\sqrt{n})$ factor in $\mathrm{poly}(n)$ space. On the other hand, if we are interested in outputting a solution $\boldsymbol{x}$, then we show that $(1+\varepsilon)$-approximations require $\Omega(d)$ space for $p>1$, and in general, β-approximations require $\tilde{\Omega}(d/\beta^{2q})$ space for $p>1$. We complement these lower bounds with the first sublinear space upper bounds for this problem, showing that we can output a β-approximation using space only $\mathrm{poly}(n)\cdot\tilde{O}(d/\beta^q)$ for $p>1$, as well as a $\sqrt{n}$-approximation using $\mathrm{poly}(n,\log d)$ space for $p=1$.

1 INTRODUCTION

When faced with an *underdetermined* linear system $\boldsymbol{Ax}=\boldsymbol{b}$ for an $n\times d$ matrix $\boldsymbol{A}$ and a n-dimensional vector $\boldsymbol{b}$, a common approach towards obtaining useful solutions is to seek a vector $\boldsymbol{x}=[x_1,\ldots,x_d]^\top$

*Part of this work done while D. Woodruff was at the Simons Institute for the Theory of Computing. D. Woodruff also acknowledges a Simons Investigator Award, NSF CCF-2335412, and Office of Naval Research award number N000142112647.

that minimizes some measure of cost. A popular choice is to minimize the least squares cost of $\boldsymbol{x}$, i.e., the ℓ_2 norm of $\boldsymbol{x}$, in which case the exact minimizer can be written in closed form as $\arg\min_{\boldsymbol{Ax}=\boldsymbol{b}} \|\boldsymbol{x}\|_2 = \boldsymbol{A}^\top(\boldsymbol{AA}^\top)^{-1}\boldsymbol{b}$. Another well-studied choice is to minimize the ℓ_1 norm of $\boldsymbol{x}$, also known as *basis pursuit* (Chen et al., 2001), which gives rise to sparse solutions $\boldsymbol{x}$ and has been popularized in the literature of compressed sensing and sparse recovery.

The minimum ℓ_1 and ℓ_2 norm solutions when $\boldsymbol{A}, \boldsymbol{b}$ are training data and labels, respectively, have been of interest to understanding the double descent phenomenon: when machine learning models exhibit multiple phases of decreasing risk (or expected loss) as model complexity increases (Hastie et al., 2022; Bartlett et al., 2020; Li & Wei, 2021). Less theory is developed about regression models in this underconstrained regime as it diverges from the traditional bias-variance U-shape risk curve. Following this framework, these interpolating models should contain very high risk as it drastically "overfits" to train data. Nonetheless, the aforementioned works demonstrate both empirical and theoretical results about the extraordinary ability of these minimum norm interpolators to generalize well on unseen data. This motivates their study in models for processing large data sets.

When $\boldsymbol{A}$ is the incidence matrix of a graph with n vertices and d edges, i.e., $\boldsymbol{A}_{u,e} = -\boldsymbol{A}_{v,e} = 1$ whenever e is an edge from a vertex u to v, then solutions $\boldsymbol{x}$ to the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ correspond to *flows* that respect a given set of *flow conservation constraints* or *demands* specified by the vector $\boldsymbol{b}$. In this case, the problem

$$\min_{\boldsymbol{Ax}=\boldsymbol{b}} \|\boldsymbol{x}\|_p \tag{1}$$

is known as the *p-norm flow problem* and has applications to graph clustering (Liu & Gleich, 2020; Fountoulakis et al., 2020), network science (Kalantari et al., 2008), and captures transshipment ($p = 1$), electrical flows ($p = 2$), and max flow ($p = \infty$) as special cases. In general, problem (1) is known as the ℓ_p regression problem, and algorithms for solving it, both for graphs and for general matrices, have recently been a topic of intense study (Bubeck et al., 2018; Ene & Vladu, 2019; Adil et al., 2019b;a; Adil & Sachdeva, 2020; Adil et al., 2021; Jambulapati et al., 2022; 2024).

It is useful to generalize problem (1) by associating a weight c_j with each column $j \in [d]$. This captures p-norm flows with associated *capacities*. In this setting, we wish to solve the optimization problem given by

$$\min_{\boldsymbol{Ax}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p \tag{2}$$

for $\boldsymbol{C} = \mathrm{diag}(c_1, \ldots, c_d)$. If $c_j = 0$ for some $j \in [d]$, then we say the objective $\|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p$ is infinite, unless $x_j = 0$.

In many underdetermined linear systems, the number d of columns of $\boldsymbol{A}$ far exceeds the number n of rows. In this regime, it may be unreasonable to store the entire matrix $\boldsymbol{A}$ in memory to solve the ℓ_p regression problem. This motivates a streaming model of computation for this problem, in which the algorithm only accesses a small portion of the input at a time in a stream of updates, and the goal is to solve the problem using memory that is much smaller than the length d of the stream. In this work, we study the *column-arrival streaming model*, defined below.

Definition 1.1 (Column-arrival streaming model)**.** Consider an instance of problem (2) given by $\boldsymbol{C} = \mathrm{diag}(c_1, \ldots, c_d) \in \mathbb{R}^{d\times d}$, $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, and $\boldsymbol{b} \in \mathbb{R}^n$. We say that this instance is presented in a *column-arrival stream* if we receive $d+1$ updates in a stream in an arbitrary order, where each update consists of either a pair $(\boldsymbol{a}^j, c_j)$ consisting of a column $\boldsymbol{a}^j$ of $\boldsymbol{A}$ and an associated weight c_j, or the vector $\boldsymbol{b}$. In this work, we focus on algorithms which make only *one pass* through the data stream.

Special cases of the general streaming underdetermined ℓ_p regression problems have been studied by a number of works in the literature of streaming algorithms. For $p = 2$, the work of Bartan & Pilanci (2023) gives an algorithm for estimating the cost of the regression problem by using sketching techniques. When $\boldsymbol{A}$ corresponds to the incidence matrix of a graph, this model corresponds to an *edge insertion graph stream* (McGregor, 2014), and the corresponding graph streaming problems for transshipment (Becker et al., 2021), electrical flows (Kelner & Levin, 2013; Kapralov et al., 2014; Cohen et al., 2016), and max flow (Assadi et al., 2019) have received much attention in the literature. The literature on graph streaming problems related to the max flow problem, such as reachability (Guruswami & Onak, 2013; Assadi & Raz, 2020; Chen et al., 2021) and maximum bipartite matching (Feigenbaum et al., 2005; McGregor, 2005; Ahn & Guha, 2013; Crouch & Stubbs, 2014; Paz & Schwartzman, 2017; Assadi et al., 2017; Kapralov, 2021) is even more vast.

In the study of streaming algorithms, we are typically interested in minimizing the space used by the algorithm. In our algorithms, we will allow for $\boldsymbol{A}$ to take real values and bound the space complexity in terms of the number of columns of $\boldsymbol{A}$ stored and additional words of space, and also give bit complexity bounds when the entries of $\boldsymbol{A}$ are bounded integers. In our lower bounds, we will prove bit complexity lower bounds in the latter complexity model.

1.1 Our Contributions

We design streaming algorithms for problem (2), handling a wide range of parameter settings. We also prove several corresponding lower bounds. We consider both the *cost estimation* problem, where the desired output is a real number that approximates the minimum ℓ_p norm, as well as the harder *vector-valued* problem, where the desired output is an approximate minimizer vector $\boldsymbol{x}$. We work in the setting $n \ll d$. In stating our results, we use the notations $\tilde{O}(\cdot)$ and $\tilde{\Omega}(\cdot)$ to suppress factors polylogarithmic in n and d.

1.1.1 Estimating the Minimum Cost

For the most basic setting of problem (1), where $p = 2$, there is a well-known closed form expression for the minimizer vector: $\boldsymbol{x}^* = \boldsymbol{A}^\top \boldsymbol{M}^{-1}\boldsymbol{b}$, where $\boldsymbol{M} := \boldsymbol{A}\boldsymbol{A}^\top$. Therefore, the minimum cost can be computed in $O(n^2)$ words of space, assuming real arithmetic. The method is straightforward: maintain the matrix $\boldsymbol{M}$ exactly, using the fact that each stream update—i.e., each new column of $\boldsymbol{A}$—makes a rank-1 additive update to $\boldsymbol{M}$. At the end of the stream, return the solution $\|\boldsymbol{A}^\top \boldsymbol{M}^{-1}\boldsymbol{b}\|_2 = (\boldsymbol{b}^\top \boldsymbol{M}^{-1}\boldsymbol{b})^{1/2}$.

Notice that the space bound $O(n^2)$ is sublinear for $d = \omega(n)$. We shall eventually prove that this quadratic dependence on n is *optimal*, even for returning a constant-factor approximation to the minimum cost. This essentially settles the streaming complexity of ℓ_p regression for $p = 2$. However, it is unclear how these results generalize to the setting of $p \neq 2$, when no closed form solutions are available for problem (1).

Our algorithms for more general $p \in (1, \infty]$ provide approximations to the cost of ℓ_p regression, up to a $(1+\varepsilon)$ factor. They are based on the streaming construction of a notion of ℓ_p *flow sparsifiers*, which we define as follows.

Definition 1.2 (Flow sparsifier)**.** Let $\boldsymbol{C} \in \mathbb{R}^{d\times d}$ be a diagonal matrix, $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, and $\boldsymbol{b} \in \mathbb{R}^n$. Let $p \in [1, \infty]$. Then, a diagonal matrix $\boldsymbol{S}$ is a β*-approximate* ℓ_p *flow sparsifier* with sparsity $\mathsf{nnz}(\boldsymbol{S})$ if

$$\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p \leq \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{S}^{-1}\boldsymbol{C}^{-1}\boldsymbol{x}\|_p \leq \beta \cdot \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p \,.$$

In the graph setting, when $\boldsymbol{A}$ is the incidence matrix of a graph, our notion of a flow sparsifier corresponds to a weighted subgraph whose ℓ_p flow cost approximates that of the original graph. This generalizes the notion of spectral sparsifiers (Spielman & Teng, 2011), which correspond to the case $p = 2$. Note that this differs from another line of work on "flow sparsifiers" that focuses on *vertex sparsification* for preserving the cost of flows when there are only a small number of terminals of interest (Leighton & Moitra, 2010; Andoni et al., 2014; Krauthgamer & Mosenzon, 2023; Chen & Tan, 2024), whereas we construct *edge* sparsifiers with general vectors $\boldsymbol{b}$, similar to a notion of flow sparsifiers used in Sherman (2013); Kelner et al. (2014).

Due to the work of Cohen et al. (2016), it is known how to construct electrical flow sparsifiers even in an *online fashion*, that is, the sparsifier can be constructed in one pass over an insertion stream of edges with the guarantee the edges are only ever kept and never thrown away.

To state the space requirements of our algorithm in full generality, we need the following notion of an *online condition number* (Cohen et al., 2016; Woodruff & Yasuda, 2022).

Definition 1.3 (Online condition number)**.** Let $\boldsymbol{A} \in \mathbb{R}^{n\times d}$. Then, the *online condition number* $\kappa^{\mathsf{OL}}(\boldsymbol{A})$ is defined as $\kappa^{\mathsf{OL}}(\boldsymbol{A}) := \|\boldsymbol{A}\| \max_{j\in[d]} \|\boldsymbol{A}_j^-\|$, where $\boldsymbol{A}_j$ denotes the $n \times j$ submatrix of $\boldsymbol{A}$ formed by its first j columns, $\boldsymbol{A}_j^-$ denotes the Moore–Penrose pseudoinverse of this matrix, and $\|\cdot\|$ denotes the matrix 2-norm.

This brings us to our main algorithmic result for constructing ℓ_p flow sparsifiers, in particular solving the cost estimation version of problem (2).

Theorem 1. *Let $p \in (1,\infty]$ and let $q = p/(p-1) \in [1,\infty)$ be its Hölder conjugate exponent. There is an algorithm that reads $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and the diagonal matrix $\boldsymbol{C} \in \mathbb{R}^{d\times d}$ in a column-arrival stream (Definition 1.1) and, with probability at least $1-\delta$, outputs a $(1+\varepsilon)$-approximate ℓ_p flow sparsifier (Definition 1.2). Furthermore, the algorithm stores at most s columns of $\boldsymbol{AC}$, with*

$$s = O\big(\varepsilon^{-2} n^{\max\{1,q/2\}}\big) \operatorname{poly}\log(d, \kappa^{\mathsf{OL}}(\boldsymbol{AC}), 1/\delta)$$

and at most $O(n^2)$ additional words of space if $\boldsymbol{A}$ has real entries, and

$$s = O\big(\varepsilon^{-2} n^{\max\{1,q/2\}}\big) \operatorname{poly}\log(d, 1/\delta)$$

and at most $O(n^2 \log d)$ bits of additional space if $\boldsymbol{AC}$ has integer entries bounded by $\pm\operatorname{poly}(d)$. Furthermore, the s columns of $\boldsymbol{AC}$ are stored in an online *fashion, that is, the columns are selected irrevocably and are not thrown away throughout the stream.*

A polylogarithmic dependence on the online condition number is typical of results in the literature of online numerical linear algebra, and is known to be necessary in general (Cohen et al., 2016). For $p \in [2,\infty]$, our algorithms store only $\tilde{O}(\varepsilon^{-2}n)$ columns of $\boldsymbol{A}$, which corresponds to an $\tilde{O}(\varepsilon^{-2}n)$ space algorithm—i.e., a "semi-streaming" algorithm—in the setting of ℓ_p flows on undirected graphs. Theorem 1 generalizes results on spectral sparsification in the semi-streaming setting (Kelner & Levin, 2013; Kapralov et al., 2014; Cohen et al., 2016), which corresponds to the case of $p = 2$. We also note that Theorem 1 separates the space complexity of max flow on undirected graphs from that on directed graphs. Indeed, on directed graphs, even the reachability problem, which is a special case of a directed max flow, requires an almost quadratic $\Omega(n^{2-o(1)})$ bits of space (Guruswami & Onak, 2013; Assadi & Raz, 2020; Chen et al., 2021), even with multiple passes.

We now turn to lower bound results. These results all have the following structure. Suppose that there exists a column-arrival streaming algorithm for problem (1) that uses at most B bits of space and, with probability at least $2/3$, outputs a "good estimate" c to the cost $\min_{\boldsymbol{Ax}=\boldsymbol{b}} \|\boldsymbol{x}\|_p$. Then, in particular, there must exist randomized algorithms $\mathscr{A}$ and $\mathscr{B}$ such that, for any $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, $\mathscr{A}$ produces an at-most-B-bit message $M = \mathscr{A}(\boldsymbol{A})$ so that, for any $\boldsymbol{b} \in \mathbb{R}^n$, $\mathscr{B}$ outputs the good estimate $c = \mathscr{B}(M, \boldsymbol{b})$ as a function of M and $\boldsymbol{b}$. The proofs of our lower bounds apply to this more relaxed setting, effectively allowing an unlimited amount of space to process the columns of $\boldsymbol{A}$ as they are streamed in and only enforcing a space limitation before $\boldsymbol{b}$ is revealed.

First, we consider the algorithmically "easy" case of problem (1), when $p = 2$. We remarked above that the folklore $O(n^2)$-space algorithm is optimal in its dependence on n. Formally, we establish the following result.

Theorem 2. *Fix $p = 2$. There is an absolute constant $\alpha > 0$ such that any column-arrival streaming algorithm that, with probability at least $2/3$, computes a $(1+\alpha)$-approximation to the cost of problem (1) requires $\Omega(n^2)$ bits of space.*

Next, we consider $p = 1$ in problem (1). In this case, the Hölder conjugate $q = \infty$ and thus Theorem 1 does not give a $(1\pm\varepsilon)$-approximation algorithm. We show that this is inherent by proving the following lower bound.

Theorem 3. *Fix $p = 1$ and let $D \geq 1$ be arbitrary. There is a constant $C_D > 0$ such that any column-arrival streaming algorithm that, with probability at least $2/3$, computes an estimate c with $c \leq \min_{\boldsymbol{Ax}=\boldsymbol{b}} \|\boldsymbol{x}\|_1 < (\sqrt{n}/C_D)c$ requires $\Omega(n^D)$ bits of space. This result applies even when all entries of the input matrix $\boldsymbol{A}$ lie in $\{\pm 1\}$.*

In other words, for $p = 1$, there is no $\operatorname{poly}(n)$ space algorithm for estimating the cost of ℓ_1 regression, even up to a factor of $o(\sqrt{n})$. In contrast, we shall soon present a nearly matching algorithm that uses just $O(n)\operatorname{poly}\log d$ bits of space to output an actual solution *vector* $\boldsymbol{x} \in \mathbb{R}^d$ that achieves a $\sqrt{n}$-factor approximation.

Finally, one can take $p = 0$ in problem (1), with the natural interpretation that $\|\boldsymbol{x}\|_0 = \mathsf{nnz}(\boldsymbol{x})$. This setting does not admit a sublinear space solution, even with n as small as 2, which we note below.

Theorem 4. *Fix $p = 0$ and take $n = 2$. Any column-arrival streaming algorithm that, with probability at least $2/3$, outputs a 2-approximation to the cost minimum cost in problem (1) requires $\Omega(d)$ bits of space.*

Table 1 summarizes the above results—both upper and lower bounds—for estimating the cost of ℓ_p regression.

Range of p	Distortion	Space complexity (bits of space)	Reference
$p=2$	1	$\tilde{O}(n^2)$	Folklore
$p\in(2,\infty]$	$(1+\varepsilon)$	$\tilde{O}(\varepsilon^{-2}n^2)$	Theorem 1
$p\in(1,2)$	$(1+\varepsilon)$	$\tilde{O}(\varepsilon^{-2}n^{q/2+1})$	Theorem 1
$p=2$	$(1+\varepsilon)$	$\Omega(n^2)$	Theorem 2
$p=1$	$o(\sqrt{n})$	$n^{\omega(1)}$	Theorem 3
$p=0$	2	$\Omega(d)$	Theorem 4

Table 1: Space complexity of estimating the cost of ℓ_p regression, with $q := p/(p-1)$

1.1.2 Outputting a Good Solution

The streaming algorithm for ℓ_p flows mentioned in Section 1.1.1 only output a *scalar*, approximating the cost of the regression problem. More generally, one would want an actual *solution vector* $\boldsymbol{x}\in\mathbb{R}^d$ with small ℓ_p norm that satisfies $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$. At first glance, this may feel like asking for too much: after all, $\boldsymbol{x}$ is d-dimensional, which would seem to necessitate at least $\Omega(d)$ space, thus precluding any sublinear space streaming algorithm. However, for $p=1$, one of the primary reasons for solving the basis pursuit problem $\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}}\|\boldsymbol{x}\|_1$ is to identify sparse solutions $\boldsymbol{x}$, which require much less than d space to specify. Furthermore, it may be possible to specify solutions in less than d space if we allow for some large distortion (i.e., approximation factor) in the solution, say $d^{0.1}$.

Thus, it *is* worthwhile to look for algorithms that can output a solution vector $\boldsymbol{x}$, especially if we allow for approximation errors as large as $\mathrm{poly}(n,d)$. We do design such an algorithm. Before discussing it, we bring up a couple of related lower bounds that we prove; these serve to set the context for the algorithm. Our lower bounds hold even in the very special setting of $n=1$, so the matrix $\boldsymbol{A}$ becomes a row vector $\boldsymbol{a}$ and the vector $\boldsymbol{b}$ becomes a scalar b. Furthermore, the entries of $\boldsymbol{a}$ can be restricted to $\{\pm1\}$ and the scalar b on the right-hand side of problem (1) can be fixed to d.

As with the lower bounds in Section 1.1.1, the next two lower bounds hold in the more relaxed setting where $\boldsymbol{a}$ can be processed by a randomized algorithm $\mathscr{A}$ using unlimited space, resulting in a B-bit message $M=\mathscr{A}(\boldsymbol{a})$. Another randomized algorithm $\mathscr{B}$ must then produce a good output vector $\hat{\boldsymbol{x}}=\mathscr{B}(M)$, based on M, succeeding with high probability. Clearly, this setting is a relaxation of a column-arrival streaming algorithm that uses B bits of space.

Theorem 5. *Let $p\in(1,\infty]$ and let $q=p/(p-1)\in[1,\infty)$ be its Hölder conjugate exponent. Let $\varepsilon\in(0,1/(8q))$ and $d\in\mathbb{N}$. Any randomized algorithm that computes a B-bit summary of $\boldsymbol{a}\in\{\pm1\}^d$ from which $\hat{\boldsymbol{x}}\in\mathbb{R}^d$ can be produced such that, with probability at least $2/3$, we have $\langle\boldsymbol{a},\hat{\boldsymbol{x}}\rangle=d$ and $\|\hat{\boldsymbol{x}}\|_p\le(1+\varepsilon)\min_{\langle\boldsymbol{a},\boldsymbol{x}\rangle=d}\|\boldsymbol{x}\|_p$ requires $B=\Omega(d)$.*

In other words, for $p>1$, there is no algorithm using less than d space that can output a $(1+\varepsilon)$-approximate solution for $\varepsilon\le1/(8q)$.

We also obtain a lower bound in the setting of large distortions, showing that for a β-factor approximation, the streaming algorithm must use at least $\tilde{\Omega}(d/\beta^{2q})$ bits of space, provided that $\beta^{3q}\ll d$. In particular, for $p>1$, it is in fact not possible to output a solution $\boldsymbol{x}$ in $\mathrm{poly}(n,\log d)$ space unless the approximation factor is at least $\mathrm{poly}(d)$.

Theorem 6. *Let $p\in(1,\infty]$ and let $q=p/(p-1)=[1,\infty)$ be its Hölder conjugate exponent. Let β be a distortion parameter such that $(\beta\log d)^{3q}=cd$ for a sufficiently small universal constant c. Then any randomized algorithm that computes a B-bit summary of $\boldsymbol{a}\in\{\pm1\}^d$ from which $\hat{\boldsymbol{x}}\in\mathbb{R}^d$ can be produced such that, with probability at least $1-1/O(\beta\log d)^q$, we have $\langle\boldsymbol{a},\hat{\boldsymbol{x}}\rangle=d$ and $\|\hat{\boldsymbol{x}}\|_p\le\beta\cdot\min_{\langle\boldsymbol{a},\boldsymbol{x}\rangle=d}\|\boldsymbol{x}\|_p$ requires $B=\Omega(d/(\beta\log d)^{2q})$.*

We prove both of these lower bounds in Section 4. The high-accuracy lower bound of Theorem 5 follows from a relatively simple reduction to one-pass streaming lower bounds for the INDEX communication problem (Kremer et al., 1995). On the other hand, Theorem 6 is our most technically advanced lower bound result, requiring additional techniques in order to extract information about $\boldsymbol{a}$ from a β-approximate solution for large β. In particular, our lower bound argument involves classifying the entries of a β-approximate solution $\boldsymbol{x}$ according to their contribution towards partitioning

the coordinates into comparable classes. We then apply conditioning on an additional short string of advice to construct an estimator that extracts many bits of information about the input $\boldsymbol{a}$ from the solution $\boldsymbol{x}$.

Turning to upper bounds, we give a new algorithm for the general ℓ_p regression problem (2) that runs in a strongly sublinear $d^{1-\Omega(1)}$ amount of space and outputs a solution vector achieving $d^{1-\Omega(1)}$ distortion. Furthermore, in the case of $p = 1$, our algorithm outputs a solution with distortion at most $\sqrt{n}$ with space complexity $\mathrm{poly}(n, \log d)$, which explains why the lower bounds of Theorems 5 and 6 do not apply when $p = 1$.

Below, we write $\boldsymbol{A}|^S$ to denote the $n \times |S|$ submatrix of $\boldsymbol{A}$ with columns indexed by $S \subseteq [d]$.

Theorem 7. *Let $p \in [1, \infty]$ and let $q = p/(p-1) \in [1, \infty]$ be its Hölder conjugate exponent. There is an algorithm that reads $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ and the diagonal matrix $\boldsymbol{C} \in \mathbb{R}^{d \times d}$ in a column-arrival stream and stores a subset S of at most $O(sd/\beta^q)$ columns of $\boldsymbol{A}$ and entries of $\boldsymbol{C}$ for $p > 1$, and $O(s)$ columns of $\boldsymbol{A}$ and entries of $\boldsymbol{C}$ for $p = 1$, such that*

$$\min_{\boldsymbol{A}|^S \boldsymbol{x} = \boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p \leq \begin{cases} \beta \cdot \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p, & \text{when } p \geq 2, \\ n^{1/p-1/2}\beta \cdot \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p, & \text{when } p < 2, \\ n^{1/2} \cdot \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p, & \text{when } p = 1, \end{cases} \tag{3}$$

where $s = O(n \log(d\kappa^{\mathsf{OL}}(\boldsymbol{AC})))$ if $\boldsymbol{AC}$ has real entries, and $s = O(n \log d)$ if $\boldsymbol{AC}$ is an integer matrix with entries bounded in absolute value by $\mathrm{poly}(d)$. *Furthermore, the s columns of $\boldsymbol{AC}$ are stored in an* online *fashion, that is, the columns are selected irrevocably and are not thrown away throughout the stream.*

As the statement of the above theorem suggests, our algorithm is based on a *column subset selection* approach. It in fact outputs a *sparse solution* that approximately solves the ℓ_p regression problem. Furthermore, the trade-off between the approximation factor β and the sparsity of this solution improves as $p \to 1$, which affirms the intuition that minimizing the ℓ_p norm for p closer to 1 yields sparser solutions. Our algorithm uses the idea of *well-conditioned spanning sets*, which is a subset of the columns of $\boldsymbol{A}$ such that all other columns can be written as a linear combination of this subset with small coefficients, and we show that such subsets can in fact be constructed in a streaming fashion by using a streaming Löwner–John ellipsoid algorithm of Woodruff & Yasuda (2022).

Table 2 summarizes our upper and lower bounds for the problem of outputting a solution vector.

Range of p	Distortion	Space complexity (bits of space)	Theorem
$p \in (1, \infty]$	$(1+\varepsilon)$	$\Omega(d)$	Theorem 5
$p \in (1, \infty]$	β	$\tilde{\Omega}(d/\beta^{2q})$	Theorem 6
$p \in (2, \infty]$	β	$n^2 \cdot \tilde{O}(d/\beta^q)$	Theorem 7
$p \in (1, 2)$	$n^{1/p-1/2}\beta$	$n^2 \cdot \tilde{O}(d/\beta^q)$	Theorem 7
$p = 1$	$n^{1/2}$	$n^2 \cdot \mathrm{poly} \log d$	Theorem 7

Table 2: Space complexity of outputting a solution vector for ℓ_p regression, with $q := p/(p-1)$

1.2 Open Questions

We have initiated the study of equality-constrained norm minimization in the streaming setting, and there are a number of natural questions that remain unresolved. Our first question is on closing the gap between our upper and lower bounds for streaming ℓ_p regression algorithms that output an approximate solution $\boldsymbol{x}$ with large distortion. We conjecture that our lower bound is tight and raise the question of tightening the upper bound.

Question 1.4. *Is there a column-arrival streaming algorithm which outputs a solution $\hat{\boldsymbol{x}}$ satisfying $\boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{b}$ and $\|\hat{\boldsymbol{x}}\|_p \leq \beta \cdot \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{x}\|_p$ using space at most* $\mathrm{poly}(n) \cdot \tilde{O}(d/\beta^{2q})$*?*

Furthermore, in this work, we have focused on the setting of one-pass algorithms, and we leave the question of studying algorithms and lower bounds for multi-pass algorithms for future work.

Question 1.5. *What is the space complexity of ℓ_p regression in the column-arrival streaming model for algorithms that make multiple passes over the data stream?*

Lastly, our study has been mostly focused on the setting of ℓ_p regression, and we have left open the possibility of obtaining better algorithms and lower bounds in the more specific setting of ℓ_p flows on undirected graphs. For instance, we already know that our lower bound of Theorem 3 can be circumvented by using streaming algorithms for constructing spanners (Althöfer et al., 1993; Feigenbaum et al., 2008; Baswana, 2008).

Question 1.6. *What is the space complexity of ℓ_p flows on undirected graphs in the edge insertion graph streaming model? What about for multi-pass algorithms?*

2 Flow Sparsifiers via Online Lewis Weight Sampling

In this section, we establish our first algorithmic result (Theorem 1). Our starting point for constructing ℓ_p flow sparsifiers is the following duality lemma; we provide a proof in Appendix B for completeness.

Lemma 2.1 (Strong duality for ℓ_p regression)**.** *Let $p \in [1,\infty]$, $q = p/(p-1) \in [1,\infty]$, and $\boldsymbol{C}$, $\boldsymbol{A}$ and $\boldsymbol{b}$ be an instance of problem (2) such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is feasible. Then*

$$\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p = \max_{\|\boldsymbol{C}\boldsymbol{A}^\top \boldsymbol{y}\|_q \le 1} \boldsymbol{y}^\top \boldsymbol{b}.$$

Proof of Theorem 1. Suppose $\boldsymbol{S} \in \mathbb{R}^{d\times d}$ is a diagonal matrix with $\|\boldsymbol{S}\boldsymbol{C}\boldsymbol{A}^\top \boldsymbol{y}\|_q = (1\pm\varepsilon)\|\boldsymbol{C}\boldsymbol{A}^\top \boldsymbol{y}\|_q$ simultaneously for every $\boldsymbol{y} \in \mathbb{R}^n$. (This notion of ℓ_q norm preservation for a subspace is known as an ℓ_q *subspace embedding*.) From the above duality lemma, it follows that a modified dual problem with capacities $\boldsymbol{S}\boldsymbol{C}$ approximates the original dual up to a $(1\pm\varepsilon)$ factor. To be precise,

$$\max_{\|\boldsymbol{S}\boldsymbol{C}\boldsymbol{A}^\top \boldsymbol{y}\|_q \le 1} \boldsymbol{y}^\top \boldsymbol{b} = (1\pm\varepsilon) \max_{\|\boldsymbol{C}\boldsymbol{A}^\top \boldsymbol{y}\|_q \le 1} \boldsymbol{y}^\top \boldsymbol{b}.$$

In turn, by applying strong duality on both sides, we have that

$$\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{S}^{-1}\boldsymbol{C}^{-1}\boldsymbol{x}\|_p = (1\pm\varepsilon) \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p. \tag{4}$$

That is, the ℓ_p flow problem with capacities $\boldsymbol{S}\boldsymbol{C}$ approximates the one capacities $\boldsymbol{C}$.

It remains to construct a sparse reweighting matrix $\boldsymbol{S}$ such that $\|\boldsymbol{S}\boldsymbol{C}\boldsymbol{A}^\top \boldsymbol{y}\|_q = (1\pm\varepsilon)\|\boldsymbol{C}\boldsymbol{A}^\top \boldsymbol{y}\|_q$ simultaneously for every $\boldsymbol{y} \in \mathbb{R}^n$. This can be done via a random sampling technique known as ℓ_p *Lewis weight sampling* (Cohen & Peng, 2015; Woodruff & Yasuda, 2023a), which shows that it is possible to construct $\boldsymbol{S}$ with

$$\mathsf{nnz}(\boldsymbol{S}) = \tilde{O}\big(\varepsilon^{-2} n^{\max\{1,q/2\}}\big). \tag{5}$$

In fact, this construction can be done even in the streaming setting via *online* ℓ_p Lewis weight sampling (Woodruff & Yasuda, 2023a), up to a small overhead, which is a factor polylogarithmic in some natural parameters. To make this precise, we recall a result of Woodruff & Yasuda (2023a) on constructing ℓ_p subspace embeddings: they show how to obtain the guarantee $\|\boldsymbol{S}\boldsymbol{B}\boldsymbol{y}\|_q = (1\pm\varepsilon)\|\boldsymbol{B}\boldsymbol{y}\|_q$ for every $\boldsymbol{y} \in \mathbb{R}^n$ when $\boldsymbol{B} \in \mathbb{R}^{d\times n}$ is presented in a *row-arrival* stream. Fortunately, this corresponds to a column-arrival stream for $\boldsymbol{A}$ when $\boldsymbol{B} = \boldsymbol{A}^\top$. Their result, translated into a column-arrival setting, is as follows.

Fact 2.2 (Online ℓ_p Lewis weight sampling, Theorem 3.8 of Woodruff & Yasuda (2023a))**.** *Let $q \in [1,\infty)$. Let $\boldsymbol{A} \in \mathbb{R}^{n\times d}$. Then, there is a column-arrival streaming algorithm which, with probability at least $1-\delta$, outputs a sampling matrix $\boldsymbol{S}$ which stores at most s rows such that for all $\boldsymbol{y} \in \mathbb{R}^n$, $\|\boldsymbol{S}\boldsymbol{A}^\top \boldsymbol{y}\|_q = (1\pm\varepsilon)\|\boldsymbol{A}^\top \boldsymbol{y}\|_q$ with*

$$s = O(\varepsilon^{-2} n^{\max\{1,q/2\}}) \operatorname{poly}\log(d, \kappa^{\mathsf{OL}}(\boldsymbol{A}), 1/\delta)$$

and at most $O(n^2)$ additional words of space if $\boldsymbol{A}$ has real entries, and

$$s = O(\varepsilon^{-2} n^{\max\{1,q/2\}}) \operatorname{poly}\log(d, 1/\delta)$$

and at most $O(n^2 \log d)$ bits of additional space if $\boldsymbol{A}$ has integer entries bounded by $\pm\operatorname{poly}(d)$.

Combining the above result with the duality-based derivation of eq. (4), we obtain our claimed algorithm for constructing ℓ_p flow sparsifiers. This completes the proof of Theorem 1 $\square$

Remark 2.3. We note that for ℓ_p regression for $p \in [2,\infty]$, our duality-based approach for estimating the regression cost is also compatible with the *turnstile streaming model*, in which the matrix $\boldsymbol{A}$ undergoes entrywise updates of the form $\boldsymbol{A}(i,j) \leftarrow \boldsymbol{A}(i,j) + \Delta$ for some update Δ which can be positive or negative. Indeed, a long line of work has established algorithms for recovering a sketch $\tilde{\boldsymbol{A}}$ such that $\|\tilde{\boldsymbol{A}}^\top \boldsymbol{y}\|_q = (1 \pm \varepsilon)\|\boldsymbol{A}^\top \boldsymbol{y}\|_q$ simultaneously for every $\boldsymbol{y} \in \mathbb{R}^n$ using only $\mathrm{poly}(n, \varepsilon^{-1}, \log d)$ bits of space (Sohler & Woodruff, 2011; Meng & Mahoney, 2013; Woodruff & Zhang, 2013; Wang & Woodruff, 2019; Li et al., 2021; Woodruff & Yasuda, 2023b; Mai et al., 2023; Munteanu & Omlor, 2024). It is also possible to get strongly sublinear in d space for $p \in (1,2)$ (i.e., $q \in (2,\infty)$) (Woodruff & Zhang, 2013). Unfortunately, these techniques yield dense approximations and thus typically do not give subquadratic space algorithms in the setting of graph streams.

Remark 2.4. If we give up the requirement that the columns are selected in an online fashion, then we can in fact remove the $\log \kappa^{\mathsf{OL}}(\mathbf{A})$ dependence from this result by using the merge-and-reduce technique to compute ℓ_p subspace embeddings (c.f. Braverman et al. (2020)).

3 Outputting a Good Solution in Sublinear Space

Here, we prove our second algorithmic result (Theorem 7). Recall that $\boldsymbol{A}|^S$ denotes the $n \times |S|$ submatrix of $\boldsymbol{A}$ with columns indexed by $S \subseteq [d]$.

3.1 Streaming Well-Conditioned Spanning Subsets

We first note that the online John ellipsoid algorithm of Woodruff & Yasuda (2022) can be used to design a streaming algorithm for identifying a small subset of rows such that every other row can be written as a linear combination of the subset with small coefficients.

Definition 3.1 (Well-conditioned spanning subsets)**.** Let $\boldsymbol{A} \in \mathbb{R}^{n \times d}$. Then, a subset $S \subseteq [d]$ of the columns of $\boldsymbol{A}$ is a *well-conditioned spanning subset* if for every $j \in [d]$, there exists $\boldsymbol{y} \in \mathbb{R}^{|S|}$ such that $\boldsymbol{A}|^S \boldsymbol{y} = \boldsymbol{a}^j$ and $\|\boldsymbol{y}\|_2^2 \leq 1$.

Algorithms for computing well-conditioned spanning subsets have been studied by Knuth (1985); Woodruff & Yasuda (2023b); Bhaskara et al. (2023). The following result gives an efficient construction of well-conditioned spanning subsets in a column-arrival stream.

Lemma 3.2 (Streaming well-conditioned spanning subsets)**.** *Let $S \subseteq [d]$ be constructed in a column-arrival stream by adding the column $\boldsymbol{a}^j$ to S whenever $(\boldsymbol{a}^j)^\top (\boldsymbol{A}|^S (\boldsymbol{A}|^S)^\top)^- \boldsymbol{a}^j \geq 1$. Then, S is a well-conditioned spanning subset. Furthermore, $|S| = O(n \log(d\kappa^{\mathsf{OL}}(\boldsymbol{A})))$ where $\kappa^{\mathsf{OL}}(\boldsymbol{A})$ is the online condition number of $\boldsymbol{A}$ (see Definition 1.3), and $|S| = O(n \log d)$ if $\boldsymbol{A}$ is an integer matrix with entries bounded by* $\mathrm{poly}(d)$.

Proof. The bound on the size of S is given by Woodruff & Yasuda (2022) using bounds on sum of online leverage scores (Cohen et al., 2016) so it remains to argue the correctness, which is inspired by an argument of Woodruff & Yasuda (2023b). If $j \in S$, then we can simply take $\boldsymbol{y}$ to be the standard basis vector corresponding to this index. Otherwise, we have that $(\boldsymbol{a}^j)^\top (\boldsymbol{A}|^S (\boldsymbol{A}|^S)^\top)^- \boldsymbol{a}^j < 1$, which means that $\boldsymbol{y} = (\boldsymbol{A}|^S)^- \boldsymbol{a}^j$ is a set of coefficients such that $\boldsymbol{A}|^S \boldsymbol{y} = \boldsymbol{a}^j$ with ℓ_2 norm at most 1. $\square$

The following lemma uses well-conditioned spanning subsets to sparsify linear combinations.

Lemma 3.3 (Sparsifying linear combinations)**.** *Let $p \in [1,\infty]$. Let $\boldsymbol{C} \in \mathbb{R}^{d \times d}$ be a diagonal matrix, $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, and let $S \subseteq [d]$ be a well-conditioned subset of the columns of $\boldsymbol{AC}$. Then, for every $\boldsymbol{x} \in \mathbb{R}^d$, there is a $\boldsymbol{z} \in \mathbb{R}^d$ such that* $\mathrm{supp}(\boldsymbol{z}) \subseteq S$, $\boldsymbol{Ax} = \boldsymbol{Az}$, *and*

$$\|\boldsymbol{z}\|_p \leq \begin{cases} d^{1-1/p}\|\boldsymbol{x}\|_p, & \text{when } p > 2, \\ n^{1/p-1/2} d^{1-1/p}\|\boldsymbol{x}\|_p, & \text{when } p \leq 2. \end{cases}$$

Proof. We may assume that $d \geq n$, since otherwise we can take $\boldsymbol{z} = \boldsymbol{x}$. By the definition of well-conditioned spanning subsets, we can write $\boldsymbol{A} = (\boldsymbol{AC})|^S (\boldsymbol{C}|^S)^{-1} \boldsymbol{Y} = \boldsymbol{A}|^S \boldsymbol{Y}$ where $\boldsymbol{A}|^S$ denotes the

columns of $\boldsymbol{A}$ indexed by S and $\boldsymbol{Y}$ is an $|S| \times d$ matrix of coefficients with $\|(\boldsymbol{C}|^S)^{-1}\boldsymbol{Y}\boldsymbol{e}^{(j)}\|_2^2 \leq 1$ for every column $j \in [d]$. Then, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}|^S\boldsymbol{Y}\boldsymbol{x}$ so padding the vector $\boldsymbol{Y}\boldsymbol{x}$ with zeros gives our vector $\boldsymbol{z}$ that is supported on S. Furthermore,

$$\begin{aligned}\|\boldsymbol{C}^{-1}\boldsymbol{z}\|_p = \|(\boldsymbol{C}|^S)^{-1}\boldsymbol{Y}\boldsymbol{x}\|_p &\leq \sum_{j=1}^{d} \|(\boldsymbol{C}|^S)^{-1}\boldsymbol{Y}\boldsymbol{e}^{(j)}\|_p |x_j| && \text{triangle inequality} \\ &\leq n^{\max\{(1/p-1/2),0\}}\|\boldsymbol{x}\|_1 \leq n^{\max\{(1/p-1/2),0\}} d^{1-1/p}\|\boldsymbol{x}\|_p. \end{aligned}$$

□

Remark 3.4. For $p \in (1,2)$ the $n^{1/p-1/2}$ factor in the distortion in Lemma 3.3 can be improved by using constructions of ℓ_p *volumetric spanners*, which give an analogue of well-conditioned spanning sets that bound the ℓ_p norm of the coefficients rather than the ℓ_2 norm (Bhaskara et al., 2023). However, this requires a larger $O(n^{q/2})$ subset size S and we do not have good streaming algorithms for constructing these objects. Thus, we do not incorporate this trade-off for the sake of simplicity.

3.2 Space-Distortion Trade-offs via Blocking

By combining the results of Lemmas 3.2 and 3.3, we immediately obtain an $O(n^2 \log d)$ space algorithm with a $\text{poly}(n,d)$ distortion. To prove Theorem 7, we will now show how to apply this in a blockwise fashion to obtain a smooth trade-off between the space complexity and the distortion.

Proof of Theorem 7. For a block size parameter $B = \min\{\beta^q, d\} \geq n$, suppose that we run the well-conditioned spanning subset algorithm of Lemma 3.2 in each of the $O(d/B)$ blocks of B columns. This algorithm stores $O(sd/B)$ columns, as we store at most s columns for each of the d/B blocks. Let S denote the set of stored columns, and let $\boldsymbol{x}^* := \arg\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \|\boldsymbol{C}^{-1}\boldsymbol{x}\|_p$. For the b-th block, let $(\boldsymbol{x}^*)^b$ denote the restriction of $\boldsymbol{x}^*$ to the b-th block of columns. Then by Lemma 3.3, $\boldsymbol{A}(\boldsymbol{x}^*)^b = \boldsymbol{A}(\hat{\boldsymbol{x}})^b$ for some $(\hat{\boldsymbol{x}})^b$ that is supported on the columns stored by the algorithm and satisfies

$$\|\boldsymbol{C}^{-1}(\hat{\boldsymbol{x}})^b\|_p \leq \begin{cases} B^{1-1/p}\|\boldsymbol{C}^{-1}(\boldsymbol{x}^*)^b\|_p\,, & \text{when } p \geq 2\,, \\ n^{1/p-1/2}B^{1-1/p}\|\boldsymbol{C}^{-1}(\boldsymbol{x}^*)^b\|_p\,, & \text{when } p < 2\,. \end{cases}$$

By concatenating these solutions across the $O(d/B)$ blocks, we obtain a solution $\hat{\boldsymbol{x}}$ that is supported on the columns stored by the algorithm, satisfies $\boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{b}$, and satisfies equation 3. □

4 Lower Bounds

Due to space constraints, the proofs of most of our lower bound results (Theorems 2, 3, 4, and 5) are given in Appendix C. However, we give the proof of Theorem 6 here, as it is technically the most interesting and illustrates several important ideas. We first state several standard facts from information theory, which can be found in, e.g., Cover & Thomas (2001).

Definition 4.1. Let X, Y, Z be random variables supported on sets $\mathscr{X}, \mathscr{Y}, \mathscr{Z}$ with probability laws p_X, p_Y, p_Z, respectively. Then, the entropy of X is defined as $\mathbf{H}(X) := \sum_{x\in\mathscr{X}} p_X(x)\log_2 \frac{1}{p_X(x)}$ and the conditional entropy of Y given X is defined as $\mathbf{H}(Y \mid X) := \sum_{x\in\mathscr{X}, y\in\mathscr{Y}} p_X(x)\cdot p_{Y|X=x}(y)\log_2 \frac{1}{p_{Y|X=x}(y)}$. The mutual information between X and Y is defined as $\mathbf{I}(X;Y) = \mathbf{H}(Y) - \mathbf{H}(Y \mid X)$ and the conditional mutual information between X and Y given Z is defined as $\mathbf{I}(X;Y \mid Z) = \mathbf{H}(Y \mid Z) - \mathbf{H}(Y \mid X, Z)$.

Fact 4.2. *For random variables X and Y, we have $\mathbf{H}(X,Y) = \mathbf{H}(X) + \mathbf{H}(Y \mid X) = \mathbf{H}(Y) + \mathbf{H}(X \mid Y)$.*

Fact 4.3 (Chain rule)**.** *For random variables $X_1, \ldots, X_d$ and Y, we have $\mathbf{H}(X_1, \ldots, X_d \mid Y) = \sum_{j=1}^{d} \mathbf{H}(X_j \mid Y, X_{<j})$, where $X_{<j}$ denotes the random variables $\{X_{j'} : j' \in [d], j' < j\}$.*

Fact 4.4. *Let X be a random variable supported on $\mathscr{X}$. Then, $\mathbf{H}(X) \leq \log_2|\mathscr{X}|$, with equality achieved when X distributed uniformly on $\mathscr{X}$.*

Fact 4.5 (Data processing inequality)**.** *Let X, Y be random variables supported on $\mathscr{X}, \mathscr{Y}$ and let $f : \mathscr{Y} \to \mathscr{X}$ be a function. Let $Z = f(Y)$. Then, $\mathbf{I}(X;Y) \geq \mathbf{I}(X;Z)$.*

Fact 4.6 (Fano's inequality)**.** *Let X, Y be random variables supported on $\mathscr{X}, \mathscr{Y}$ and let $f : \mathscr{Y} \to \mathscr{X}$ be a function. Let $\tilde{X} = f(\mathscr{Y})$ and let E denote the event that $X \neq \tilde{X}$. Then, $\mathbf{H}(X \mid Y) \leq \mathrm{H}_b(\mathbf{Pr}[E]) + \mathbf{Pr}[E]\cdot\log(|\mathscr{X}|-1)$ where $\mathrm{H}_b(x) := -x\log_2 x - (1-x)\log_2(1-x)$ is the entropy of a Bernoulli random variable with parameter $x \in [0,1]$*

Next we prove our lower bound for large approximation factors.

Theorem 6. *Let $p \in (1,\infty]$ and let $q = p/(p-1) = [1,\infty)$ be its Hölder conjugate exponent. Let β be a distortion parameter such that $(\beta \log d)^{3q} = cd$ for a sufficiently small universal constant c. Then any randomized algorithm that computes a B-bit summary of $\boldsymbol{a} \in \{\pm 1\}^d$ from which $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ can be produced such that, with probability at least $1 - 1/O(\beta \log d)^q$, we have $\langle \boldsymbol{a}, \hat{\boldsymbol{x}} \rangle = d$ and $\|\hat{\boldsymbol{x}}\|_p \leq \beta \cdot \min_{\langle \boldsymbol{a}, \boldsymbol{x} \rangle = d} \|\boldsymbol{x}\|_p$ requires $B = \Omega(d/(\beta \log d)^{2q})$.*

Proof. Let $\boldsymbol{a} \in \{\pm 1\}^d$ be a random sign vector. Suppose that Alice constructs a message M as a function of $\boldsymbol{a}$, sends the message to Bob, and Bob constructs $\boldsymbol{x}$ such that $\langle \boldsymbol{a}, \hat{\boldsymbol{x}} \rangle = d$ with $\|\hat{\boldsymbol{x}}\|_p \leq \beta \cdot \min_{\langle \boldsymbol{a}, \boldsymbol{x} \rangle = d} \|\boldsymbol{x}\|_p$. Since we can take $\boldsymbol{x} = \boldsymbol{a}$, the optimal solution must have ℓ_p norm at most $d^{1/p}$.

We will lower bound the length of the message M by obtaining a lower bound on the mutual information $\mathbf{I}(\hat{\boldsymbol{x}}\,;\boldsymbol{a})$ between Alice's random vector $\boldsymbol{a}$ and Bob's solution $\hat{\boldsymbol{x}}$. Indeed, the entropy of the message $\mathbf{H}(M)$ lower bounds the length of the message, and we have $\mathbf{I}(\hat{\boldsymbol{x}}\,;\boldsymbol{a}) \leq \mathbf{I}(M\,;\boldsymbol{a}) \leq \mathbf{H}(M)$ by the data processing inequality.

Let $\boldsymbol{y}$ be obtained by rounding the entries of $\hat{\boldsymbol{x}}$ to the nearest integer, so that $\langle \boldsymbol{a}, \boldsymbol{y} \rangle \geq d/2$ and $\boldsymbol{y}$ takes on at most $K := O(\beta \cdot d^{1/p})$ distinct values. Now consider partitioning $[K]$ into $\lceil \log_2 K \rceil$ groups, where the ℓ-th group is given by $C_\ell = \{k \in [K] : 2^{\ell-1} \leq k \leq 2^\ell\}$. By averaging, there is a group ℓ^* such that $\sum_{k \in C_{\ell^*}} d_k \geq \frac{1}{\lceil \log_2 K \rceil} \sum_{k=1}^K d_k \geq \frac{d}{2\lceil \log_2 K \rceil}$. Note that there must be at least $2^{-\ell^*} d/(2\lceil \log_2 K \rceil)$ coordinates belonging to the class C_{ℓ^*}, so we must have that $(2^{\ell^*-1})^p 2^{-\ell^*} \frac{d}{2\lceil \log_2 K \rceil} \leq \|\boldsymbol{y}\|_p^p \leq 2\beta^p d$ and thus we have that $2^{\ell^*} \leq L := O(\beta^q (\log d)^{q/p})$.

For each $k \in [K]$, let $G_k \subseteq [d]$ denote the subset of coordinates for which $y_j \in \{\pm k\}$, let $\boldsymbol{y}^k$ denote the restriction of $\boldsymbol{y}$ to G_k, and let $d_k = \langle \boldsymbol{a}, \boldsymbol{y}^k \rangle$. Let $D = \{\mathrm{sign}(d_k)\}_{k=1}^L$. Then, $\mathbf{I}(\hat{\boldsymbol{x}}\,;\boldsymbol{a}) = \mathbf{I}(\hat{\boldsymbol{x}}\,;\boldsymbol{a} \mid D) + \mathbf{I}(\hat{\boldsymbol{x}}\,;D) - \mathbf{I}(\hat{\boldsymbol{x}}\,;D \mid \boldsymbol{a}) \geq \mathbf{I}(\hat{\boldsymbol{x}}\,;\boldsymbol{a} \mid D) - \mathbf{H}(D) \geq \mathbf{I}(\hat{\boldsymbol{x}}\,;\boldsymbol{a} \mid D) - L$. Furthermore, we have

$$\begin{aligned}\mathbf{I}(\hat{\boldsymbol{x}}\,;\boldsymbol{a} \mid D) \geq \mathbf{I}(\boldsymbol{y}\,;\boldsymbol{a} \mid D) &= \sum_{j=1}^d \mathbf{I}(\boldsymbol{y}\,;a_j \mid \boldsymbol{a}_{<j}, D) = \sum_{j=1}^d \mathbf{H}(a_j \mid \boldsymbol{a}_{<j}, D) - \mathbf{H}(a_j \mid \boldsymbol{y}, \boldsymbol{a}_{<j}, D) \\ &\geq \mathbf{H}(\boldsymbol{a} \mid D) - \sum_{j=1}^d \mathbf{H}(a_j \mid \boldsymbol{y}, D)\,. \end{aligned} \tag{6}$$

By Fact 4.2, the first term is bounded by $\mathbf{H}(\boldsymbol{a} \mid D) = \mathbf{H}(\boldsymbol{a}) - \mathbf{H}(D) + \mathbf{H}(D \mid \boldsymbol{a}) \geq d - L$ so it remains the bound the latter term. We do this by constructing an estimator $\hat{\boldsymbol{a}}$ for $\boldsymbol{a}$ and then applying Fano's inequality. Our estimator $\hat{\boldsymbol{a}}$ is given by $\hat{a}_j = \mathrm{sign}(d_k)\,\mathrm{sign}(y_j)$ if the j-th coordinate has value $|y_j| = k \leq L$, and a random sign otherwise. We will now bound $\mathbf{Pr}[\hat{a}_j \neq a_j]$.

For each k, let J_k be the number of coordinates $j \in G_k$ such that $\hat{a}_j = a_j$. We then have $|d_k| = |\langle \boldsymbol{a}, \boldsymbol{y}^k \rangle| = k \cdot (J_k - (|G_k| - J_k)) = k \cdot (2J_k - |G_k|)$ so the probability that $\hat{a}_j = a_j$ for a coordinate $j \in G_k$ is $\frac{J_k}{|G_k|} = \frac{1}{2} + \frac{|d_k|}{2k|G_k|}$. Then, conditioned on the success of the algorithm and a choice of the G_k and d_k, the probability that $\hat{a}_j = a_j$ is $\sum_{k=1}^L \frac{J_k}{d}$, which equals

$$\sum_{k=1}^L \frac{|G_k|}{d}\left(\frac{1}{2} + \frac{|d_k|}{2k|G_k|}\right) = \frac{1}{2} + \frac{1}{d}\sum_{k=1}^L \frac{|d_k|}{2k} \geq \frac{1}{2} + \frac{1}{dL}\sum_{k \in C_{\ell^*}} |d_k| \geq \frac{1}{2} + \frac{1}{2L\lceil \log_2 K \rceil} = \frac{1}{2} + \frac{1}{O(\beta \log d)^q}\,.$$

Then overall, we have that

$$\mathbf{Pr}[\hat{a}_j \neq a_j] \geq \left(1 - \frac{1}{O(\beta \log d)^q}\right)\left(\frac{1}{2} + \frac{1}{O(\beta \log d)^q}\right) \geq \frac{1}{2} + \frac{1}{O(\beta \log d)^q}\,.$$

By Fano's inequality (Fact 4.6), we then have that $\mathbf{H}(a_j \mid \hat{a}_j) \leq \mathrm{H}_b(\mathbf{Pr}[\hat{a}_j \neq a_j]) \leq 1 - 1/O(\beta \log d)^{2q}$. Now plugging into eq. (6) gives that $\mathbf{I}(\boldsymbol{x}\,;\boldsymbol{a} \mid D) \geq d - L - (d - O(d/(\beta \log d)^{2q})) = \Omega(d/(\beta \log d)^{2q}) - L$ and thus $\mathbf{I}(\boldsymbol{x}\,;\boldsymbol{a}) \geq \Omega(d/(\beta \log d)^{2q}) - 2L = \Omega(d/(\beta \log d)^{2q})$ for L small enough as required by the theorem hypothesis. Thus, the length of the message M sent by Alice must be at least $\Omega(d/(\beta \log d)^{2q})$ bits, which concludes the proof. □

ACKNOWLEDGMENTS

Funding agency acks go here. We thank the anonymous reviewers for useful feedback on improving the presentation of this work. David P. Woodruff and Taisuke Yasuda were supported by a Simons Investigator Award.

BIBLIOGRAPHY

P.-A. Absil, A. Edelman, and P. Koev. On the largest principal angle between random subspaces. *Linear Algebra Appl.*, 414(1):288–294, 2006. ISSN 0024-3795. doi: 10.1016/j.laa.2005.10.004. URL https://doi.org/10.1016/j.laa.2005.10.004. C.3

Deeksha Adil and Sushant Sachdeva. Faster p-norm minimizing flows, via smoothed q-norm problems. In Shuchi Chawla (ed.), *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pp. 892–910. SIAM, 2020. doi: 10.1137/1.9781611975994.54. URL https://doi.org/10.1137/1.9781611975994.54. 1

Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Iterative refinement for ℓ_p-norm regression. In Timothy M. Chan (ed.), *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pp. 1405–1424. SIAM, 2019a. doi: 10.1137/1.9781611975482.86. URL https://doi.org/10.1137/1.9781611975482.86. 1

Deeksha Adil, Richard Peng, and Sushant Sachdeva. Fast, provably convergent IRLS algorithm for p-norm linear regression. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14166–14177, 2019b. URL https://proceedings.neurips.cc/paper/2019/hash/46c7cb50b373877fb2f8d5c4517bb969-Abstract.html. 1

Deeksha Adil, Brian Bullins, Rasmus Kyng, and Sushant Sachdeva. Almost-linear-time weighted ℓ_p-norm solvers in slightly dense graphs via sparsification. In Nikhil Bansal, Emanuela Merelli, and James Worrell (eds.), *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pp. 9:1–9:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi: 10.4230/LIPIcs.ICALP.2021.9. URL https://doi.org/10.4230/LIPIcs.ICALP.2021.9. 1

Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013. doi: 10.1016/J.IC.2012.10.006. URL https://doi.org/10.1016/j.ic.2012.10.006. 1

Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discret. Comput. Geom.*, 9:81–100, 1993. doi: 10.1007/BF02189308. URL https://doi.org/10.1007/BF02189308. 1.2

Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards $(1+\varepsilon)$-approximate flow sparsifiers. In Chandra Chekuri (ed.), *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pp. 279–293. SIAM, 2014. doi: 10.1137/1.9781611973402.20. URL https://doi.org/10.1137/1.9781611973402.20. 1.1.1

Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In Sandy Irani (ed.), *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pp. 342–353. IEEE, 2020. doi: 10.1109/FOCS46700.2020.00040. URL https://doi.org/10.1109/FOCS46700.2020.00040. 1, 1.1.1

Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In Philip N. Klein (ed.), *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pp. 1723–1742. SIAM, 2017. doi: 10.1137/1.9781611974782.113. URL https://doi.org/10.1137/1.9781611974782.113. 1

Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In Moses Charikar and Edith Cohen (eds.), *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pp. 265–276. ACM, 2019. doi: 10.1145/3313276.3316361. URL https://doi.org/10.1145/3313276.3316361. 1

Burak Bartan and Mert Pilanci. Distributed sketching for randomized optimization: Exact characterization, concentration, and lower bounds. *IEEE Trans. Inf. Theory*, 69(6):3850–3879, 2023. doi: 10.1109/TIT.2023.3247559. URL https://doi.org/10.1109/TIT.2023.3247559. 1

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. volume 117, pp. 30063–30070. National Acad Sciences, 2020. 1

Surender Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 106(3):110–114, 2008. doi: 10.1016/J.IPL.2007.11.001. URL https://doi.org/10.1016/j.ipl.2007.11.001. 1.2

Ruben Becker, Sebastian Forster, Andreas Karrenbauer, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. *SIAM J. Comput.*, 50(3):815–856, 2021. doi: 10.1137/19M1286955. URL https://doi.org/10.1137/19M1286955. 1

Aditya Bhaskara, Sepideh Mahabadi, and Ali Vakilian. Tight bounds for volumetric spanners and applications. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/02a92b52670752daf17b53f04f1ab405-Abstract-Conference.html. 3.1, 3.4

Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pp. 517–528. IEEE, 2020. doi: 10.1109/FOCS46700.2020.00055. URL https://doi.org/10.1109/FOCS46700.2020.00055. 2.4

Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, and Yuanzhi Li. An homotopy method for ℓ_p regression provably beyond self-concordance and in input-sparsity time. In Ilias Diakonikolas, David Kempe, and Monika Henzinger (eds.), *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pp. 1130–1137. ACM, 2018. doi: 10.1145/3188745.3188776. URL https://doi.org/10.1145/3188745.3188776. 1

Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Almost optimal super-constant-pass streaming lower bounds for reachability. In Samir Khuller and Virginia Vassilevska Williams (eds.), *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pp. 570–583. ACM, 2021. doi: 10.1145/3406325.3451038. URL https://doi.org/10.1145/3406325.3451038. 1, 1.1.1

Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, 2001. doi: 10.1137/S003614450037906X. URL https://doi.org/10.1137/S003614450037906X. 1

Yu Chen and Zihan Tan. On $(1+\varepsilon)$-approximate flow sparsifiers. In David P. Woodruff (ed.), *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pp. 1568–1605. SIAM, 2024. doi: 10.1137/1.9781611977912.63. URL https://doi.org/10.1137/1.9781611977912.63. 1.1.1

Michael B. Cohen and Richard Peng. L_p row sampling by lewis weights. In Rocco A. Servedio and Ronitt Rubinfeld (eds.), *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pp. 183–192. ACM, 2015. doi: 10.1145/2746539.2746567. URL https://doi.org/10.1145/2746539.2746567. 2, A

Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In Tim Roughgarden (ed.), *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pp. 181–190. ACM, 2015. doi: 10.1145/2688073.2688113. URL https://doi.org/10.1145/2688073.2688113. A

Michael B. Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans (eds.), *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPIcs*, pp. 7:1–7:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi: 10.4230/LIPIcs.APPROX-RANDOM.2016.7. URL https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2016.7. 1, 1.1.1, 1.1.1, 3.1, A, A, A

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2001. ISBN 9780471062592. doi: 10.1002/0471200611. URL https://doi.org/10.1002/0471200611. 4

Michael S. Crouch and Daniel M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore (eds.), *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *LIPIcs*, pp. 96–104. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. doi: 10.4230/LIPICS.APPROX-RANDOM.2014.96. URL https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.96. 1

Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pp. 1127–1136. ACM Press, 2006. URL http://dl.acm.org/citation.cfm?id=1109557.1109682. A

Alina Ene and Adrian Vladu. Improved convergence for ℓ_1 and ℓ_∞ regression via iteratively reweighted least squares. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1794–1801. PMLR, 2019. URL http://proceedings.mlr.press/v97/ene19a.html. 1

Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi: 10.1016/J.TCS.2005.09.013. URL https://doi.org/10.1016/j.tcs.2005.09.013. 1

Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM J. Comput.*, 38(5):1709–1727, 2008. doi: 10.1137/070683155. URL https://doi.org/10.1137/070683155. 1.2

Kimon Fountoulakis, Di Wang, and Shenghao Yang. p-norm flow diffusion for local graph clustering. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3222–3232. PMLR, 2020. URL http://proceedings.mlr.press/v119/fountoulakis20a.html. 1

Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM J. Comput.*, 45(5):1762–1792, 2016. doi: 10.1137/15M1009718. URL https://doi.org/10.1137/15M1009718. C.3

Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pp. 287–298. IEEE Computer Society, 2013. doi: 10.1109/CCC.2013.37. URL https://doi.org/10.1109/CCC.2013.37. 1, 1.1.1

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. volume 50, pp. 949. NIH Public Access, 2022. 1

Arun Jambulapati, Yang P. Liu, and Aaron Sidford. Improved iteration complexities for overconstrained p-norm regression. In Stefano Leonardi and Anupam Gupta (eds.), *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pp. 529–542. ACM, 2022. doi: 10.1145/3519935.3519971. URL https://doi.org/10.1145/3519935.3519971. 1

Arun Jambulapati, James R. Lee, Yang P. Liu, and Aaron Sidford. Sparsifying generalized linear models. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell (eds.), *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pp. 1665–1675. ACM, 2024. doi: 10.1145/3618260.3649684. URL https://doi.org/10.1145/3618260.3649684. 1

Mehdi Kalantari, Masoumeh Haghpanahi, and Mark A. Shayman. A p-norm flow optimization problem in dense wireless sensor networks. In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*, pp. 341–345. IEEE, 2008. doi: 10.1109/INFOCOM.2008.77. URL https://doi.org/10.1109/INFOCOM.2008.77. 1

Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In Dániel Marx (ed.), *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pp. 1874–1893. SIAM, 2021. doi: 10.1137/1.9781611976465.112. URL https://doi.org/10.1137/1.9781611976465.112. 1

Michael Kapralov and Kunal Talwar. On differentially private low rank approximation. In Sanjeev Khanna (ed.), *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pp. 1395–1414. SIAM, 2013. doi: 10.1137/1.9781611973105.101. URL https://doi.org/10.1137/1.9781611973105.101. C.3

Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pp. 561–570. IEEE Computer Society, 2014. doi: 10.1109/FOCS.2014.66. URL https://doi.org/10.1109/FOCS.2014.66. 1, 1.1.1

Jonathan A. Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory Comput. Syst.*, 53(2):243–262, 2013. doi: 10.1007/S00224-012-9396-1. URL https://doi.org/10.1007/s00224-012-9396-1. 1, 1.1.1

Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In Chandra Chekuri (ed.), *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pp. 217–226. SIAM, 2014. doi: 10.1137/1.9781611973402.16. URL https://doi.org/10.1137/1.9781611973402.16. 1.1.1

Donald E Knuth. Semi-optimal bases for linear dependencies. *Linear and Multilinear Algebra*, 17 (1):1–4, 1985. 3.1

Robert Krauthgamer and Ron Mosenzon. Exact flow sparsification requires unbounded size. In Nikhil Bansal and Viswanath Nagarajan (eds.), *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pp. 2354–2367. SIAM, 2023. doi: 10.1137/1.9781611977554.CH91. URL https://doi.org/10.1137/1.9781611977554.ch91. 1.1.1

Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. In Frank Thomson Leighton and Allan Borodin (eds.), *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pp. 596–605. ACM, 1995. doi: 10.1145/225058.225277. URL https://doi.org/10.1145/225058.225277. 1.1.2

Frank Thomson Leighton and Ankur Moitra. Extensions and limits to vertex sparsification. In Leonard J. Schulman (ed.), *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pp. 47–56. ACM, 2010. doi: 10.1145/1806689.1806698. URL https://doi.org/10.1145/1806689.1806698. 1.1.1

Yi Li, David P. Woodruff, and Taisuke Yasuda. Exponentially improved dimensionality reduction for ℓ_1: Subspace embeddings and independence testing. In Mikhail Belkin and Samory Kpotufe (eds.), *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pp. 3111–3195. PMLR, 2021. URL http://proceedings.mlr.press/v134/li21c.html. 2.3

Yue Li and Yuting Wei. Minimum ℓ_1-norm interpolators: Precise asymptotics and multiple descent. 2021. 1

Meng Liu and David F. Gleich. Strongly local p-norm-cut algorithms for semi-supervised learning and local graph clustering. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3501672ebc68a5524629080e3ef60aef-Abstract.html. 1

Tung Mai, Alexander Munteanu, Cameron Musco, Anup Rao, Chris Schwiegelshohn, and David P. Woodruff. Optimal sketching bounds for sparse linear regression. In Francisco J. R. Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent (eds.), *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pp. 11288–11316. PMLR, 2023. URL https://proceedings.mlr.press/v206/mai23a.html. 2.3

Andrew McGregor. Finding graph matchings in data streams. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan (eds.), *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pp. 170–181. Springer, 2005. doi: 10.1007/11538462_15. URL https://doi.org/10.1007/11538462_15. 1

Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. doi: 10.1145/2627692.2627694. URL https://doi.org/10.1145/2627692.2627694. 1

Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (eds.), *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pp. 91–100. ACM, 2013. doi: 10.1145/2488608.2488621. URL https://doi.org/10.1145/2488608.2488621. 2.3

Alexander Munteanu and Simon Omlor. Turnstile ℓ_p leverage score sampling with applications. In *Proceedings of the 41st International Conference on Machine Learning, ICML 2024*, Proceedings of Machine Learning Research. PMLR, 2024. 2.3

Udaya Parampalli, Xiaohu Tang, and Serdar Boztas. On the construction of binary sequence families with low correlation and large sizes. *IEEE Trans. Inf. Theory*, 59(2):1082–1089, 2013. doi: 10.1109/TIT.2012.2219691. URL https://doi.org/10.1109/TIT.2012.2219691. 8

Ami Paz and Gregory Schwartzman. A $(2+\varepsilon)$-approximation for maximum weight matching in the semi-streaming model. In Philip N. Klein (ed.), *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pp. 2153–2161. SIAM, 2017. doi: 10.1137/1.9781611974782.140. URL https://doi.org/10.1137/1.9781611974782.140. 1

Anup Rao and Amir Yehudayoff. *Communication complexity and applications*. Cambridge University Press, Cambridge, 2020. ISBN 978-1-108-49798-5. C.1

Jonah Sherman. Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pp. 263–269. IEEE Computer Society, 2013. doi: 10.1109/FOCS.2013.36. URL https://doi.org/10.1109/FOCS.2013.36. 1.1.1

Christian Sohler and David P. Woodruff. Subspace embeddings for the l_1-norm with applications. In Lance Fortnow and Salil P. Vadhan (eds.), *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pp. 755–764. ACM, 2011. doi: 10.1145/1993636.1993736. URL https://doi.org/10.1145/1993636.1993736. 2.3

Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011. doi: 10.1137/08074489X. URL https://doi.org/10.1137/08074489X. 1.1.1

Ruosong Wang and David P. Woodruff. Tight bounds for ℓ_p oblivious subspace embeddings. In Timothy M. Chan (ed.), *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pp. 1825–1843. SIAM, 2019. doi: 10.1137/1.9781611975482.110. URL https://doi.org/10.1137/1.9781611975482.110. 2.3

David P. Woodruff and Taisuke Yasuda. High-dimensional geometric streaming in polynomial space. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pp. 732–743. IEEE, 2022. doi: 10.1109/FOCS54457.2022.00075. URL https://doi.org/10.1109/FOCS54457.2022.00075. 1.1.1, 1.1.2, 3.1, 3.1, C.3

David P. Woodruff and Taisuke Yasuda. Online Lewis weight sampling. In Nikhil Bansal and Viswanath Nagarajan (eds.), *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pp. 4622–4666. SIAM, 2023a. doi: 10.1137/1.9781611977554.ch175. URL https://doi.org/10.1137/1.9781611977554.ch175. 2, 2, 2.2, A

David P. Woodruff and Taisuke Yasuda. New subset selection algorithms for low rank approximation: Offline and online. In Barna Saha and Rocco A. Servedio (eds.), *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pp. 1802–1813. ACM, 2023b. doi: 10.1145/3564246.3585100. URL https://doi.org/10.1145/3564246.3585100. 2.3, 3.1, 3.1

David P. Woodruff and Qin Zhang. Subspace embeddings and ℓ_p-regression using exponential random variables. In Shai Shalev-Shwartz and Ingo Steinwart (eds.), *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, volume 30 of *JMLR Workshop and Conference Proceedings*, pp. 546–567. JMLR.org, 2013. URL http://proceedings.mlr.press/v30/Woodruff13.html. 2.3