Subspace Exploration: Bounds on Projected Frequency Estimation

Graham Cormode
University of Warwick

Charlie Dickens
University of Warwick

David P. Woodruff
Carnegie Mellon University

ABSTRACT

Given an $n \times d$ dimensional dataset A, a projection query specifies a subset $C \subseteq [d]$ of columns which yields a new $n \times |C|$ array. We study the space complexity of computing data analysis functions over such subspaces, including heavy hitters and norms, when the subspaces are revealed only after observing the data. We show that this important class of problems is typically hard: for many problems, we show $2^{\Omega(d)}$ lower bounds. However, we present upper bounds which demonstrate space dependency better than 2^d. That is, for $c, c' \in (0,1)$ and a parameter $N = 2^d$ an N^c-approximation can be obtained in space $\min(N^{c'}, n)$, showing that it is possible to improve on the naïve approach of keeping information for all 2^d subsets of d columns. Our results are based on careful constructions of instances using coding theory and novel combinatorial reductions that exhibit such space-approximation tradeoffs.

CCS CONCEPTS

• **Theory of computation** → **Streaming models**; **Lower bounds and information complexity**; **Communication complexity**; **Sketching and sampling**.

KEYWORDS

projection queries, distinct elements, frequency moments

ACM Reference Format:
Graham Cormode, Charlie Dickens, and David P. Woodruff. 2021. Subspace Exploration: Bounds on Projected Frequency Estimation. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3452021.3458312

1 INTRODUCTION

In many data analysis scenarios, datasets of interest are of moderate to high dimension, but many of these dimensions are spurious or irrelevant. Thus, we are interested in subspaces, corresponding to the data projected on a particular subset of dimensions. Within each subspace, we are concerned with computing statistics, such as norms, measures of variation, or finding common patterns. Such calculations are the basis of subsequent analysis, such as regression and clustering. In this paper, we introduce and formalize novel problems related to functions of the frequency in such projected subspaces. Already, special cases such as subspace projected distinct elements have begun to generate interest, e.g., in Vu's work [18], and as an open problem in sublinear algorithms [16].

In more detail, we consider the original data to be represented by a (usually binary) array with n rows of d dimensions. A subspace is defined by a set $C \subseteq [d]$ of columns, which defines a new array with n rows and $|C|$ dimensions. Our goal is to understand the complexity of answering queries, such as which rows occur most frequently in the projected data, computing frequency moments over the rows, and so on. If C is provided prior to seeing the data, then the projection can be performed online, and so many of these tasks reduce to previously studied questions. Hence, we focus on the case when C is decided *after* the data is seen. In particular, we may wish to try out many different choices of C to explore the structure of the subspaces of the data. Our model is given in detail in Section 2.

For further motivation, we outline some specific areas where such problems arise.

- **Bias and Diversity.** A growing concern in data analysis and machine learning is whether outcomes are 'fair' to different subgroups within the population, or whether they reinforce existing disparities. A starting point for this is to quantify the level of bias within the data when different features are considered. That is, we want to know whether certain combinations of attribute values are over-represented in the data (heavy hitters), and how many different combinations of values are represented in the data (captured by measures like F_0). We would like to be able to answer such queries accurately for many different (typically overlapping) subsets of dimensions.
- **Privacy and Linkability.** When sharing datasets, we seek assurance that they are not vulnerable to attacks that exploit structure in the data to re-identify individuals. An attempt to quantify this risk is given in recent work [6], which asks how many distinct values occur in the data for each partial identifier, specified as a subset of dimensions. This prior work considered the case where the target dimensions are known in advance, but more generally we would like to compute such measures for arbitrary subsets, based on frequency moments and sampling techniques.
- **Clustering and Frequency Analysis.** In the area of clustering, the notion of subspaces has been studied under a number of interpretations. The common theme is that the data may look unclustered in the original space due to spurious dimensions inflating the distance between points that are otherwise close. Many papers addressed this as a search problem: to search through exponentially many subspaces to find those in which the data is well-clustered. See the survey by Parsons, Haque and Liu [15]. In our setting, the

problem would be to estimate various measures of density or clusteredness for a given subspace. A related problem is to find subspaces (or "subcubes" in database terminology) that have high frequency. Prior work proceeded under strong statistical independence assumptions about the values in different dimensions, for example, that the distribution can be modeled accurately with a (Naïve) Bayesian model [13].

2 PRELIMINARIES AND DEFINITIONS

For a positive integer Q, let $[Q] = \{0, 1, \ldots, Q-1\}$, and $A \in [Q]^{n\times d}$ be the input data. The objective is to keep a summary of A which is used to estimate the solution to a problem **P** upon receiving a column subset query $C \subseteq [d]$. Problems **P** of interest are described in Section 2.1. Define the restriction of A to the columns indexed by C as A^C whose rows A_i^C, $1 \le i \le n$, are vectors over $[Q]^{|C|}$. We use the Minkowski norm $\|X\|_p = (\sum_{i,j} |X_{ij}|^p)^{1/p}$ to denote the entrywise-ℓ_p norm for vectors ($j = 1$) and matrices ($j > 1$).

Computational Model. First, the data A is received under the assumption that it is too large to hold entirely in memory so can be modeled as a stream of data. Our lower bounds are not strongly dependent on the order in which the data is presented. After observing A, a *column query* C is presented. The frequency vector over A induced by C is $f = f(A, C)$ whose entries $f_i(A, C)$ denote the frequency of Q-ary word $w_i \in [Q]^{|C|}$. We study functions of the frequency vector $f = f(A, C)$ after the observation of A and receiving column query C. The task is, during the observation phase, to design a summary of A which approximates statistics of A^C, the restriction of A to its projected subspace C. Approximations of A^C are accessed through the frequency vector $f(A, C)$. Note that functions (e.g., norms) are taken over $f(A, C)$ as opposed to the raw vector inputs from the column projection.

REMARK 1 (INDEXING Q-ARY WORDS INTO f). *Recall that the frequency vector $f(A, C)$ has length $Q^{|C|}$ with each entry f_i counting the occurrences of word $w_i \in [Q]^{|C|}$. To clearly distinguish between the (scalar) index i of f and the input vectors w_i whose frequency is measured by f_i we introduce the **index function** $e(w_i) = i$. We may think of $e(\cdot)$ as simply the canonical mapping from $[Q]^{|C|}$ into $\{0, 1, 2, \ldots, Q^C - 1\}$, but other suitable bijections may be used.*

For example, suppose $Q = 2$ and $A \in \{0, 1\}^{5\times 3}$ with column indices $\{1, 2, 3\}$ given below. If $C = \{1, 2\}$, then using the canonical mapping from $\{0, 1\}^{|C|}$ into $\{0, 1, 2, 3\}$ (e.g $e(00) = 0, e(01) = 1, \ldots e(11) = 3$) we obtain A^C and hence $f(A, C) = (1, 1, 0, 3)$.

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad \longrightarrow \qquad A^C = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

The vector $f = f(A, C)$ is then the frequency vector over which we seek to compute statistical queries such as $\|f\|_0$. In this example, $\|f\|_0 = 3$ (there are three distinct rows in A^C), while $\|f\|_1 = 5$ is independent of the choice of C.

2.1 Problem Definitions.

The problems that we consider are column-projected forms of common streaming problems ([11], [3], [4]). Here, we refer to these problems as "projected frequency estimation problems" over the input A. We define

$$f_i(A, C) = |\{j : A_j^C = w_i, j \in [n]\}| \qquad (1)$$

$$F_p(A, C) = \sum_{i\in\{0,1\}^{|C|}} f_i(A, C)^p. \qquad (2)$$

- F_p **estimation**: Given a column query C, the F_p estimation problem is to approximate the quantity $F_p(A, C) = \|f(A, C)\|_p^p$ under some measure of approximation to be specified later (e.g., up to a constant factor). Of particular interest to us is (projected) $F_0(A, C)$ estimation, which counts the number of distinct row patterns in A^C.
- ℓ_p**-heavy hitters**: The query is specified by a column query $C \subseteq [d]$, a choice of metric/norm $\ell_p, p > 0$ and accuracy parameter $\phi \in (0, 1)$. The task is then to identify all patterns w_i observed on A^C for which $f_i(A, C) \ge \phi\|f(A, C)\|_p$. Such values w_i (or equivalently i) are called *ϕ-ℓ_p-heavy hitters*, or simply *ℓ_p-heavy hitters* when ϕ is fixed. We will consider a multiplicative approximation based on a parameter $c > 1$, where we require that all ϕ-ℓ_p heavy hitters are reported, and no items with weight less than $(\phi/c) \cdot \|f(A, C)\|_p$ are included.
- ℓ_p**-frequency estimation**: A related problem is to allow the frequency $f_i(A, C)$ to be estimated accurately, with error as a fraction of $F_p(A, C)^{1/p} = \|f(A, C)\|_p$, which we refer to as *$\ell_p$ frequency estimation*. Specifically, for a given w_i, return an estimate $\hat{f}_i$ which satisfies $|\hat{f}_i(A, C) - f_i(A, C)| \le \phi\|f(A, C)\|_p$.
- ℓ_p **sampling**: The goal of this sampling problem is to sample patterns w_i according to the distribution $p_i \in (1 \pm \varepsilon)\frac{f_i^p(A,C)}{\|f(A,C)\|_p^p} + \Delta$ where $\Delta = 1/\text{poly}(nd)$, and return a $(1\pm\varepsilon')$-approximation to the probability p_i of the item w_i returned.

When clear, we may drop the dependence upon C in the notation and write f_i and F_p instead. We will use $\tilde{O}$ and $\tilde{\Omega}$ notation to supress factors that are polylogarithmic in the leading term. For example, lower bounds stated as $\tilde{\Omega}(2^d)$ suppress terms polynomial in d.

2.2 Related Work

The model we study is reminscent of, but distinct from, some related formulations. In the problem of cascaded aggregates [10], we imagine the starting data as a matrix, and apply a first operator (denoted Q) on each row to obtain a vector, on which we apply a second operator P. Our problems can be understood as special cases of cascaded aggregates where Q is a project-then-concatenate operator, to obtain a vector whose indices correspond to the concatenation of the projection of a row. Another example of a cascaded aggregate is a so-called correlated aggregate [17], but this was only studied in the context of two dimensions. To the best of our knowledge, our projection-based definitions have not been previously studied under the banner of cascaded aggregates.

Other work includes results on provisioning queries for analytics [2], but the way these statistics are defined is different from our formulation. In that setting there are different scenarios ("hypotheticals") that may or may not be turned on: this corresponds to "what-if" analysis whereby a query is roughly "how many items are observed if a given set of columns is present (turned on)?" The number of distinct elements for the query is the union of the number of distinct elements across scenarios. In our setting, we concatenate the distinct items into a row vector and count the number of distinct vectors. Note that in the hypotheticals setting in the binary case, each column only has 2 distinct values, 0 and 1, and thus the union also only has 2 distinct values. However, we can obtain up to 2^d distinct vectors. Consequently, Assadi et al. are able to achieve $\mathrm{poly}(d/\varepsilon)$ space for counting distinct elements, whereas we show a $2^{\Omega(d)}$ lower bound. Moreover, they achieve a $2^{\Omega(d)}$ lower bound for counting (i.e., F_1), whereas we achieve a constant upper bound. These disparities highlight the differences in our models.

More recently, the notion of "subset norms" was introduced by Braverman, Krauthgamer and Yang [5]. This problem considers an input that defines a vector v, where the objective is to take a subset s of entries of v and compute the norm. Results are parameterized by the "heavy hitter dimension", which is a measure of complexity over the set system from which s can be drawn. While sharing some properties with our scenario, the results for this model are quite different. In particular, in [5] a trivial upper bound follows by maintaining the vector v explicitly, of dimension n. Meanwhile, many of our results show lower bounds that are exponential in the dimensionality, as $2^{\Omega(d)}$, though we also obtain non-trivial upper bounds.

3 CONTRIBUTIONS

The main challenge here is that the column query C is revealed *after* observing the data; consequently, applying a known algorithm to just the columns C as the data arrives is not possible. For example, consider the exemplar problem of counting the number of distinct rows under the projection C, i.e., the projected F_0 problem. Recall that A_i^C denotes the i-th row of array A^C. Then the task is to count the number of distinct rows observed in A^C, i.e.,

$$F_0(A,C) = |\{A_j^C : j \in [n]\}| = \|f(A,C)\|_0.$$

Observe that $F_0(A,C)$ can vary widely over different choices of C. For example, even for a binary input $A \in \{0,1\}^{n\times d}$, $F_0(A,C)$ can be as large as 2^d when C consists of all columns from a highly diverse dataset, and as small as 1 or 2 when C is a single column or when C selects homogeneous columns (e.g., the columns in C are all zeros).

3.1 Summary of Results

Our main focus, in common with prior work on streaming algorithms, is on space complexity. For the above problems we obtain the following results:

- In Section 4 we show that projected F_0 estimation requires $2^{\Omega(d)}$ space for a constant factor approximation, demonstrating the essential hardness of these problems. Nevertheless, we obtain a tradeoff in terms of upper bounds described below.
- Section 5 presents results for ℓ_p frequency estimation, ℓ_p heavy hitters, F_p estimation, and ℓ_p sampling. We show a space upper bound of $O(\varepsilon^{-2}\log(1/\delta))$ for ℓ_p frequency estimation when $0 < p < 1$ and complement this result with lower bounds for heavy hitters when $p > 1$, F_p estimation and ℓ_p sampling for all $p \neq 1$, showing that these problems require $2^{\Omega(d)}$ bits of space.
- In Section 6 we show upper bounds for F_0 and F_p estimation which improve on the exhaustive approach of keeping summaries of all 2^d subsets of columns, by showing that we can obtain coarse approximate answers with a smaller subset of materialized answers. Specifically, for parameters $N = 2^d$ and $\alpha \in (0,1)$ we can obtain an N^α approximation in $\min\left(N^{H(1/2-\alpha)}, n\right)$ space. Since the binary entropy function $H(x) < 1$, this bound is better than the trivial 2^d bound.

These bounds show that there is no possibility of "super efficient" solutions that use space less than exponential in d. Nevertheless, we demonstrate some solutions whose dependence is still exponential but weaker than a naïve 2^d. Thinking of $N = 2^d$, the above upper and lower bounds imply the actual complexity is a nontrivial polynomial function of N.

The bounds also show novel dichotomies that are not present in comparable problems without projection. In particular, we show that (projected) ℓ_p sampling is difficult for $p \neq 1$ while (projected) ℓ_p-heavy hitters has a small space algorithm for $0 < p < 1$. This differs from the standard streaming model in which the (classical) ℓ_p heavy hitters problem has a small space solutions for $p \leq 2$ without projection [14], and (classical) ℓ_p sampling can be performed efficiently for $p \leq 2$ [9]. Our lower bounds are built on amplifying the frequency of target codewords for a carefully chosen test word.

Note that there are trivial naïve solutions which simply retain the entire input and so answer the query exactly on the query C: to do so takes $\Theta(nd)$ space, noting that n may be exponential in d. Alternatively, if we know $t = |C|$ then we may enumerate all $\binom{d}{t}$ subsets of $[d]$ with size t and maintain (approximate) summaries for each choice of C. However, this will entail a cost of at least $\Omega(d^t)$ and as such does not give a major reduction in cost.

3.2 Coding Theory Definitions

Our lower bounds will typically make use of a *binary* code $\mathcal{C}$, constituted of a collection of *codewords*, which are vectors (or strings) of fixed length. We write $\mathcal{B}(l,k)$ to denote all binary strings of length l and (Hamming) weight k. We first consider the dense, low-distance family of codes $\mathcal{C} = \mathcal{B}(d,k)$ but will later use more sophisticated randomly sampled codes. When $k < d/2$, we have $\binom{d}{k} \geq (d/k)^k$ and when $k = d/2$, we have $\binom{d}{d/2} \geq 2^d/\sqrt{2d}$. A trivial but crucial property of $\mathcal{B}(d,k)$ is that any two codewords from this set can have intersecting 1s in at most $k-1$ positions.

We define the *support of a string* y as $\mathrm{supp}(y) = \{i : y_i \neq 0\}$, the set of locations where y is non-zero. We define *child words* to be the set of new codewords obtained from $\mathcal{C}$ by generating all Q-ary words z with $\mathrm{supp}(z) \subseteq \mathrm{supp}(y)$ for some $y \in \mathcal{C}$, and construct them with the star operator defined next.

Definition 3.1 (star^Q operation, child words). Let d be the length of a binary word, k be a weight parameter, and suppose $y \in \mathcal{B}(d,k)$. Let $M = \mathrm{supp}(y)$. We define the function $\mathrm{star}^Q(y)$ to be the operation which lifts a binary word y to a larger alphabet by generating all the words over alphabet $[Q]$ on M. Formally,

$$\mathrm{star}^Q(y \in \{0,1\}^d) = \{z : z \in [Q]^d, \mathrm{supp}(z) \subseteq \mathrm{supp}(y)\}$$

Since the alphabet size Q is often fixed when using this operation, when clear we will drop the superscript and abuse notation by writing $\mathrm{star}(y)$. Elements of the set $\mathrm{star}^Q(y)$ are referred to as *child words* of y.

For any $y \in \mathcal{B}(d,k)$, there are Q^k words generated by $\mathrm{star}^Q(y)$. When $\mathrm{star}(\cdot)$ is applied to all vectors of a set U then we write $\mathrm{star}(U) = \cup_{u \in U}\mathrm{star}(u)$. For example, if $y \in \{0,1\}^d$ and $Q = 2$, then $\mathrm{star}^Q(y)$ is simply all possible binary words of length d whose support is contained in $\mathrm{supp}(y)$. For the projected F_0 problem, the code $C = \mathcal{B}(d,k)$ is sufficient. However, for our subsequent results, we need a randomly chosen code whose existence is demonstrated in Lemma 3.2. The proof follows from a Chernoff bound.

Lemma 3.2. *Fix $\epsilon, \gamma \in (0,1)$ and let $C \subseteq \mathcal{B}(d, \epsilon d)$ be such that for any two distinct $x, y \in C$ we have $|x \cap y| \leq (\epsilon^2 + \gamma)d$. With probability at least $1 - \exp(-2d\gamma^2)$ there exists such a code C with size $2^{O(\gamma^2 d)}$ instantiated by sampling sufficiently many words i.i.d. at random from $\mathcal{B}(d, \epsilon d)$.*

Proof. Let X be the random variable for the number of 1s in common between x and y sampled uniformly at random. Then the expectation of X is $\mathbb{E}[X] = \frac{(\epsilon d)^2}{d} = \epsilon^2 d$ and although the coordinates of x, y are not independent, they are negatively correlated so we may use a Chernoff bound (see Section 1.10.2 of [7] for self-contained details). Our aim is to show that the number of 1s in common between x and y can be at most γd more than its expectation. Then, via an additive Chernoff-Hoeffding bound:

$$\mathbb{P}(X - \mathbb{E}(X) \geq \gamma d) \leq \exp(-2d\gamma^2).$$

This is the probability that any two codewords x and y are not too similar, so by taking a union bound over the $\Theta(|C|^2)$ pairs of codewords, the size of the code is $|C| = \exp(d\gamma^2) = 2^{\gamma^2 d/\ln 2}$. □

3.3 Overview of Lower Bound Constructions

Our lower bounds rely upon non-standard reductions to the Index problem using codes C defined in Section 3.2. These reductions are more involved than is typically found as we need to combine the combinatorial properties of C along with the $\mathrm{star}(\cdot)$ operation on Alice's input. In particular, the interplay between C and $\mathrm{star}(\cdot)$ must be understood over the column query S given by Bob, which again relies on properties of C used to define the input.

Recall that the typical reduction from Index is as follows: Alice holds a vector $\mathbf{a} \in \{0,1\}^N$, Bob holds an index $i \in [N]$ and he is tasked with finding $\mathbf{a}_i$ following one-way communication from Alice. The randomized communication complexity of Index is $\Omega(N)$ [12]. We adapt this setup for our family of problems, following an approach that has been used to prove many space lower bounds for streaming algorithms.

The general construction of our lower bounds is as follows: first we choose a binary code C (usually independently at random) with certain properties such as a specific weight and a bounded number of 1s in common locations with other words in the code. In the communication setting, Alice holds a subset $T \subseteq C$ while Bob holds a codeword $y \in C$ and is tasked with determining whether or not $y \in T$. Bob can also access the index function (Remark 1) $e(y)$ which simply returns the index or location that y is enumerated in C. The corresponding bitstring for the Index problem that Alice holds is $\mathbf{a} \in \{0,1\}^{|C|}$ which has $\mathbf{a}_j = 1$ for every element $w_j \in T$ (under a suitable enumeration of $\{0,1\}^d$). We use the $\mathrm{star}(T)$ operator (defined in Section 3.2) to map these strings into an input A for each of the problems (i.e., a collection of rows of datapoints). Upon defining the instance, we show that Bob can query a proposed algorithm for the problem and use the output to determine whether or not Alice holds y. This enables Bob to return $\mathbf{a}_{e(y)}$, which is 1 if Alice holds $y \in T$ and 0 otherwise. Hence, determining if $y \in T$ or $y \in C \setminus T$ solves Index and incurs the lower bound $\Omega(|C|)$. Our constructions of C establish that $|C|$ is exponentially large in d.

4 LOWER BOUNDS FOR F_0

In this section, we focus on the F_0 (distinct counting) projected frequency problem. The main result in this section is a strong lower bound for the problem, which is exponential in the domain size d.

We use codes $C = \mathcal{B}(d,k)$ as defined in Section 3.2.

Theorem 4.1. *Let $Q \geq 2$ be the target alphabet size and $k < d/2$ be a fixed query size with $Q > k$. Any algorithm achieving an approximation factor of $|Q|/k$ for the projected F_0 problem requires space $2^{\Omega(d)}$.*

Proof. Fix the code $C = \mathcal{B}(d,k)$, recalling that any $x \in C$ has Hamming weight k, and for distinct $x, y \in C$ at most $k-1$ bits are shared in common. We will use these facts to obtain the approximation factor.

Obtain the collection of all child words C_Q from C by using $\mathrm{star}^Q(\cdot)$ as defined in Section 3.2. We will reduce from the Index problem in communication complexity as follows. Alice has a set of (binary) codewords $T \subseteq C$ and initializes the input array A for the algorithm with all strings from the set $\mathrm{star}(T)$. Bob has a vector $y \in C$ and wants to know if $y \in T$ or not. Let $S = \mathrm{supp}(y)$ so that $|S| = k$ and Bob queries the F_0 algorithm on columns of A restricted to S. First suppose that $y \in T$. Then Alice holds y so $\mathrm{star}(y)$ is included in A and there must be at least Q^k patterns observed. Conversely, if $y \notin T$, then Alice does not include y in A. However, by the construction of C, y shares at most $(k-1)$ 1s with any distinct $y' \in C$. Thus, the number of patterns observed on the columns corresponding to S is at most $\binom{k}{k-1}Q^{k-1} = kQ^{k-1}$.

We observe that if we can distinguish the case of kQ^{k-1} from Q^k, then we could correctly answer the Index instance, i.e., if we can achieve an approximation factor of Δ such that:

$$\Delta = \frac{Q^k}{kQ^{k-1}} = \frac{Q}{k}. \tag{3}$$

Any protocol for Index requires communication proportional to the length of Alice's input vector $\mathbf{a}$, which translates into a space lower bound for our problem. Alice's set $T \subset C$ defines an input vector for the Index problem built using a characteristic vector over all words in C, denoted by $\mathbf{a} \in \{0,1\}^{|C|}$, as follows. Under a suitable enumeration of $C = \{w_1, w_2, \ldots, w_{|C|}\}$, Alice's vector is encoded via $\mathbf{a}_i = 1$ if and only if Alice holds the binary word $w_i \in T$. From the separation shown prior to (3), Bob can determine if Alice holds a word in T, thus solving Index and incurring the lower bound. Hence, space proportional to $|C| = \binom{d}{k}$ is necessary. We use the standard relation $\binom{d}{k} \geq (d/k)^k$ and choose $k = ad/2$ for a constant $a \in [0,1)$ from which we obtain $|C| \geq 2^{ad/2}$ to achieve the stated approximation guarantee. □

Setting $k = ad/2$ allows us to vary the query size and directly understand how this affects the size of the code necessary for the lower bound. For a query of size k, the size of the input to the projected F_0 problem is a matrix whose rows are words contained in $\mathrm{star}^Q(T)$, hence A has size $|T|Q^k \times d$. Theorem 4.1 is for $k < d/2$. When $k = d/2$ we can use the tighter bound for the central binomial term on the sum of the binomial coefficients and obtain the following stronger bounds. The subsequent results use the same encoding as in Theorem 4.1. However, at certain points of the calculations the parameter setttings are slightly altered to obtain different guarantees.

Corollary 4.2. *Let $Q \geq d/2$ be an alphabet size and $d/2$ be the query size. There exists a choice of input data $A \in [Q]^{n \times d}$ such that any algorithm achieving approximation factor $2Q/d$ for the projected F_0 problem on the query requires space $2^{\Omega(d)}$.*

Proof. Repeat the argument of Theorem 4.1 with $k = d/2$. The approximation factor from equation (3) becomes $\Delta = 2Q/d$. The code size for Index is $|C| \geq 2^d/\sqrt{2d}$. Note that $|C|$ is $2^{\Omega(d)}$ as $\frac{1}{2}\log_2(d)$ can always be bounded above by a linear function of d. The instance is an array whose rows are the $Q^{d/2}$ child words in $\mathrm{star}^Q(T)$. Hence, the size of the instance to the F_0 algorithm is bounded above by $|T|Q^{d/2} \times d$. □

Corollary 4.3 follows from Corollary 4.2 by setting $Q = d$.

Corollary 4.3. *A 2-factor approximation to the projected F_0 problem on a query of size $d/2$ needs space $2^{\Omega(d)}$ with an instance A whose size is $|T|Q^{d/2} \times d$.*

Theorem 4.1 and its corollaries suffice to obtain space bounds over all choices of Q. However, Q could potentially grow to be very large, which may be unsatisfying. As a result, we will argue how the error varies for fixed Q. To do so, we map $[Q]$ down to a smaller alphabet $[q]$ and use this code to define the communication problem from which the lower bound will follow. The cost of this is that the instance is a logarithmic factor larger in the dimensionality.

Corollary 4.4. *Let q be a target alphabet size such that $2 \leq q \leq |Q|$. Let $\alpha = Q\log_q(Q) \geq 1$ and $d' = d\log_q(Q)$. There exists a choice of input data $A \in [q]^{n \times d'}$ for which any algorithm for the projected F_0 problem over queries of size $d/2$ that guarantees error $\tilde{O}(\alpha/d')$ requires space $2^{\Omega(d)}$.*

Proof. Fix the binary code $C = \mathcal{B}(d, d/2)$ and generate all child words over alphabet $[Q]$ to obtain the approximation factor $\Delta = 2Q/d$ as in Corollary 4.3. For every $w \in C$ there are $Q^{d/2}$ child words so the child code C_Q now has size $n = \Theta(2^d Q^{d/2}/\sqrt{d})$ words. Since Q can be arbitrarily large, we encode it via a mapping to a smaller alphabet but over a slightly larger dimension; specifically, use a function $[Q] \mapsto [q]^{\log_q(Q)}$ which generates q-ary strings for each symbol in $[Q]$. Hence, all of the stored strings in $C_Q \subset [Q]^d$ are equivalent to a collection, C_q over $[q]^{d\log_q(Q)}$. Although $|C_Q| = |C_q|$, words in C_Q are length d, while the equivalent word in C_q has length $d\log_q(Q)$. This collection of words from C_q now defines the instance $A \in [q]^{n \times d\log_q(Q)}$, each word being a row of A. Taking $\alpha = Q\log_q(Q)$ and $d' = d\log_q(Q)$ results in an approximation factor of:

$$\Delta = \frac{2Q}{d} = \frac{2\alpha}{d'}. \tag{4}$$

Alice's input vector $\mathbf{a}$ is defined by the same code C and held set $T \subset C$ as in Theorem 4.1 so we incur the same space bound. Likewise, Bob's test vector y and column query S also remain the same as in that theorem. □

Table 1: Comparison of the lower bounds for F_0. Theorem 4.1 uses $C = \mathcal{B}(d,k)$, corollaries use $C = \mathcal{B}(d, d/2)$.

	Instance A for F_0	Approx. Factor
Theorem 4.1	$\lvert T\rvert Q^k \times d$ over $[Q]$	Q/k
Corollary 4.2	$\lvert T\rvert Q^{d/2} \times d$ over $[Q]$	$2Q/d$
Corollary 4.3	$\lvert T\rvert Q^{d/2} \times d$ over $[d]$	2
Corollary 4.4	$\lvert T\rvert Q^{d/2} \times d\log_q Q$ over $[q]$	$2Q/d$

Corollary 4.4 says that the same accuracy guarantee as Corollary 4.2 can be given by reducing the arbitrarily large alphabet $[Q]$ to a smaller one over $[q]$. However, the price to pay for this is that the size of the instance A increases by a factor of $\log_q(Q)$ in the dimensionality. These various results are summarized in Table 1.

5 ℓ_p-FREQUENCY BASED PROBLEMS

In this section, we extend the techniques from the previous section to understand the complexity of projected frequency estimation problems related to the ℓ_p norms and F_p frequency moments (defined in Section 2.1). A number of our results are lower bounds, but we begin with a simple sampling-based upper bound to set the stage.

5.1 ℓ_p Frequency Estimation

We first focus on the projected frequency estimation problem showing that a simple algorithm keeping a uniform sample of the rows works for $p < 1$. The algorithm `uSample`(A, C, t, b) first builds a uniform sample of t rows (sampled with replacement at rate $\alpha = t/n$) from A and evaluates the absolute frequency of string b on the sample after projection onto C. Let g be the absolute frequency of b on the subsample. To estimate the true frequency of b on the entire dataset from the subsample, we return an appropriately scaled estimator $\hat{f}_{e(b)} = g/\alpha$ which meets the required bounds given in Theorem 5.1, recalling that $e(b)$ is the index location associated with the string b. The proof follows by a standard Chernoff bound argument and is given in Appendix A.1.

Theorem 5.1. *Let $A \in \{0,1\}^{n \times d}$ be the input data and let $C \subseteq [d]$ be a given column query. For a given string $b \in \{0,1\}^C$, the absolute frequency of b, $f_{e(b)}$, can be estimated up to $\varepsilon\|f\|_1$ additive error using a uniform sample of size $O(\varepsilon^{-2}\log(1/\delta))$ with probability at least $1-\delta$.*

The same algorithm can be used to obtain bounds for all $0 < p < 1$. By noting that $\|f\|_1 \le \|f\|_p$ for $0 < p < 1$ we can obtain the following corollary.

Corollary 5.2. *Let A, b, C be as in Theorem 5.1. Let $0 < p < 1$. Then uniformly sampling $O(\varepsilon^{-2}\log(1/\delta))$ rows achieves*

$$\left|\hat{f}_{e(b)} - f_{e(b)}\right| \le \varepsilon\|f\|_p$$

with probability at least $1-\delta$.

Both Theorem 5.1 and Corollary 5.2 are stated as if C is given. However, since the sampling did not rely on C in any way, we can sample complete rows of the input uniformly prior to receiving the query C, which is revealed after observing the data. The uniform sampling approach also allows us to identify the ℓ_p heavy hitters in small space: for each item included in the sample (when projected onto column set C), we use the sample to estimate its frequency, and declare those with high enough estimated frequency to be the heavy hitters. By contrast, for $p > 1$ we are able to obtain a $2^{\Omega(d)}$ space lower bound, given in the next section.

5.2 ℓ_p Heavy Hitters Lower Bound

Recall from Section 2.1 that the objective of (projected) ϕ-ℓ_p heavy hitters is to find all those rows in A^C whose frequency is at least some ϕ-fraction of the ℓ_p norm of the frequency distribution of this projection. For the lower bound we need a randomly sampled code as defined in Lemma 3.2. The lower bound argument follows a similar outline to the bound for F_0, although now Bob's query is on the complement of the support of his test vector y (i.e., $S = [d] \setminus \operatorname{supp}(y)$) rather than $\operatorname{supp}(y)$. Akin to Theorem 4.1, we will create a reduction from the Index problem in communication complexity, and use its communication lower bound to argue a space lower bound for projected ℓ_p heavy hitters. The proof will generate an instance of ℓ_p heavy hitters based on encoding a collection of codewords, and consider in particular the status of the string corresponding to all zeros. We will consider two cases: when Bob's query string is represented in Alice's set of codewords, then the all zeros string will be a heavy hitter (for a subset of columns determined by the query); and when Bob's string is not in the set, then the all zeros string will not be a heavy hitter. We begin by setting up the encoding of the input to the Index instance.

Theorem 5.3. *Let $\phi \in (0,1)$ be a parameter and fix $p > 1$. Any algorithm which can obtain a constant factor approximation to the projected ϕ-ℓ_p-heavy hitters problem requires space $2^{\Omega(d)}$.*

Proof. Fix $\epsilon > 0$. Let $C \subset \mathcal{B}(d, \epsilon d)$ be a code whose words have weight ϵd and any two distinct words x, y have at most $(\epsilon^2 + \gamma)d$ ones in common. By Lemma 3.2 such a C exists and $|C| = 2^{\Omega_\gamma(d)}$.

Suppose Alice holds a subset $T \subset C$. Let $\mathbf{a} \in \{0,1\}^{|C|}$ be the characteristic vector over all length-d binary strings for which $\mathbf{a}_{e(u)} = 1$ if and only if Alice holds $u \in T$. Bob holds $y \in C$ and wants to determine if Alice holds $y \in T$. Ascertaining whether or not Alice holds y would be sufficient for Bob to solve Index and incur the $\Omega(|C|)$ lower bound.

The input array, A, for the ϕ-ℓ_p-heavy hitters problem is constructed as follows.

(1) Alice populates A with $2^{\epsilon d}$ copies of the length-d all ones vector, $\mathbf{1}_d$
(2) Next, Alice takes $Q = 2$ and inserts into A the collection $\operatorname{star}^Q(T)$, which is the expansion of her input strings to all child-words in binary. That is, for every $s \in T$, Alice computes all binary strings x of length d with $\operatorname{supp}(x) \subseteq \operatorname{supp}(s)$ and includes these in A.

Let $S = [d] \setminus \operatorname{supp}(y)$, so that $|S| = d - \epsilon d = (1-\epsilon)d$. Without loss of generality we may assume $S = \{1, 2, \ldots, (1-\epsilon)d\}$ and we denote the $(1-\epsilon)d$ length vector which is identically 0 on S by $\mathbf{0}_S$. Suppose there is an algorithm $\mathscr{A}$ which approximates the ℓ_p-heavy hitters problem on a given column query up to a constant approximation factor. Bob queries $\mathscr{A}$ for the heavy hitters in the table A under the column query given by the set S, and then uses this information to answer whether or not $y \in T$.

Case 1: $y \in T$. If $y \in T$, then we claim that $\mathbf{0}_S$ is a ϕ-ℓ_p heavy hitter for some constant ϕ, i.e., $f_{e(\mathbf{0}_S)} \ge \phi\|f\|_p$. We will manipulate the equivalent condition $f^p_{e(\mathbf{0}_S)} \ge \phi^p F_p$. Since $y \in T$, the set $\operatorname{star}(y)$ is included in the table A as Alice inserted $\operatorname{star}(s)$ for every s that she holds. Consider any child word of y, that is, a $w \in \operatorname{star}(y)$. Since y is supported only on $[d] \setminus S$ and $\operatorname{supp}(w) \subseteq \operatorname{supp}(y)$, every $w_i = 0$ for $i \in S$. So $\mathbf{0}_S$ is observed once for every $w \in \operatorname{star}(y)$ and there are $|\operatorname{star}(y)| = 2^{\epsilon d}$ such w. Hence, $\mathbf{0}_S$ occurs at least $2^{\epsilon d}$ times.

Now that we have a lower bound on the frequency of $\mathbf{0}_S$, it remains to upper bound the F_p value when $y \in T$ so that we are assured $\mathbf{0}_S$ will be a heavy hitter in this instance. The quantity we seek is the F_p value of all vectors in A^S, written $F_p(A, S)$; which we decompose into the contribution from $\mathbf{0}_S$ present due to y being in T, and two special cases from the block of $2^{\epsilon d}$ all-ones rows and 'extra' copies of $\mathbf{0}_S$ which are contributed by vectors $y' \ne y$. We claim that this $F_p(A, S)$ value is at most $|C|^{1+p}2^{\epsilon d + (\epsilon^2+\gamma)dp} + 3 \cdot 2^{\epsilon p d}$.

First, let $y' \in C$ with $y' \ne y$ and consider prefixes z supported on S which can be generated by possible child words from $\operatorname{star}(y')$. Since our code requires that $|y' \cap y| \le (\epsilon^2 + \gamma)d$, y' can have at most $(\epsilon^2+\gamma)d$ 1s located in $\bar{S} = [d] \setminus S$, and hence must have at least $(\epsilon - \epsilon^2 - \gamma)d$ 1s located in S. Since $|\operatorname{star}(y')| = 2^{\epsilon d}$, the number of copies of z inserted is at most $2^{\epsilon d - (\epsilon d - \epsilon^2 d - \gamma d)} = 2^{\epsilon^2 d + \gamma d}$. This occurs for every $y' \in C$ so the total number of occurences of z is at most $|C|2^{(\epsilon^2+\gamma)d}$. The contribution to F_p for this scenario is then $|C|^p 2^{(\epsilon^2+\gamma)dp}$. Observe that each codeword y' generates at most $2^{\epsilon d}$ vectors under the $\operatorname{star}(y')$ operator, so we have an upper bound of $|C|2^{\epsilon d}$ such vectors generated, with a total contribution of $|C|^{1+p}2^{(\epsilon^2 p + \epsilon + \gamma p)d}$.

Next, we focus on the two special vectors to count which have a high contribution to the F_p value. Recall that Alice specifically included $\mathbf{1}_d$ into A $2^{\epsilon d}$ times so the p-th powered frequency is exactly $2^{\epsilon pd}$ for this term. From the above argument, $\mathbf{0}_S$ also has frequency $2^{\epsilon d}$ from $\mathsf{star}(y)$. But $\mathbf{0}_S$ is also created at most $2^{(\epsilon^2+\gamma)d}$ times from each $y' \neq y$ in T, giving an additional count of at most $|C|2^{(\epsilon^2+\gamma)d}$. Based on our choice of ϵ and γ, we can ensure that this is asymptotically smaller than $2^{\epsilon d}$, and so the total contribution from these two special vectors is at most $3 \cdot 2^{\epsilon d}$. So in total we achieve that F_p is at most $|C|^{1+p} 2^{\epsilon d + (\epsilon^2+\gamma)dp} + 3 \cdot 2^{\epsilon pd}$, as claimed.

Then $\mathbf{0}_S$ meets the definition to be a ϕ-ℓ_p heavy hitter provided

$$2^{\epsilon pd} > \phi^p (|C|^{1+p} 2^{\epsilon d + (\epsilon^2+\gamma)pd} + 3 \cdot 2^{\epsilon pd}).$$

Assuming $p > 1$, and choosing ϵ sufficiently smaller than $(p-1)/p$ and γ sufficiently small, we have that

$$|C|^{1+p} 2^{\epsilon d + (\epsilon^2+\gamma)pd} \le 2^{O(\gamma^2 d(1+p)) + \epsilon d + \epsilon(p-1)d + \gamma pd} \le 2^{\epsilon pd}.$$

Hence, we require $2^{\epsilon pd} > \phi^p O(2^{\epsilon pd})$, i.e., $2^{\epsilon d} > \phi O(2^{\epsilon d})$, which is satisfied for a suitably small but constant ϕ.

Case 2: $y \notin T$**.** On the other hand, suppose that $y \notin T$. Then the claim is that $\mathbf{0}_S$ is not a ϕ-ℓ_p-heavy hitter. Now the vector $\mathbf{0}_S$ does not occur with a high frequency because $\mathsf{star}(y)$ is not included in A. However, certain child words in $\mathsf{star}(T)$ could also generate $\mathbf{0}_S$ when projected onto S and this is the contribution we need to upper bound. Again, any codeword $s \in T$ has at least $(\epsilon - \epsilon^2 - \gamma)d$ 1s present on S. So for a particular $s \in T$, $\mathbf{0}_S$ can occur $2^{\epsilon^2 d + \gamma d}$ times. Taken over all $y' \in C$ for which Alice includes in A, the frequency of $\mathbf{0}_S$ in this case is at most $|C|2^{\epsilon^2 d + \gamma d}$. Taking $\varepsilon < 1/3, \gamma < \varepsilon/3$ and using $|C| = 2^{\gamma^2 d/\ln 2}$ (Lemma 3.2) we have $f_{e(\mathbf{0}_S)} \le 2^{0.72\varepsilon d}$. Meanwhile, there are $2^{\epsilon d}$ copies of the string $\mathbf{1}_d$ inserted into A meaning that $F_p(A, S) \ge 2^{\epsilon pd}$ and hence $F_p^{1/p}$ is strictly greater than $f_{e(\mathbf{0}_S)}$. Hence, $\mathbf{0}_S$ is *not* a ϕ-ℓ_p heavy hitter provided that $f_{e(\mathbf{0}_S)}/F_p^{1/p} = 2^{-0.28\varepsilon d}$ is strictly less than $\phi = 1/4$, this is satisfied for suitable ε and d.

Concluding the proof. Bob can use his test vector y and a query S with a constant factor approximation algorithm $\mathcal{A}$ for the ℓ_p-heavy hitters problem and distinguish between the two cases of Alice holding y or not based on whether $\mathbf{0}_S$ is reported. As a result, Bob can determine if $y \in T$ and consequently solve Index, thus incurring the $\Omega(|C|) = 2^{\Omega(d)}$ lower bound. □

The instance A is initialized with $2^{\varepsilon d}$ rows of the vector $\mathbf{1}_d$ and the child words $\mathsf{star}^Q(T)$. For any $t \in \mathsf{star}^Q(T)$, $|\mathsf{star}^Q(t)| = 2^{\varepsilon d}$ so the size of the instance A is $(|T|+1)2^{\varepsilon d} \times d$.

5.3 F_p Estimation

The space complexity of approximating the frequency moments F_p has been widely studied since the pioneering work of Alon, Matias and Szegedy [1]. Here, we investigate their complexity under projection. For $p = 1$, the frequency is always the number n of rows in the original instance irrespective of the column set C, so only one word of space is required. We therefore devote attention to $p \neq 1$.

The reduction to Index for Theorem 5.4 follows a similar outline as Theorem 5.3 for $p > 1$. For $p < 1$, we encode the problem slightly differently, closer to that in Theorem 4.1. Again, the reduction to Index relies on Bob determining whether or not Alice holds y, which for F_p estimation amounts to Bob evaluating $F_p(A, S)$ and comparing to a threshold value.

Theorem 5.4. *Fix a real number $p > 0$ with $p \neq 1$. A constant factor approximation to the projected F_p estimation problem requires space $2^{\Omega(d)}$.*

Proof. For $p > 1$ we begin by noticing that in the proof for Theorem 5.3 one can also monitor the F_p value of the input to the problem rather than simply checking the heavy hitters. In particular, depending on whether or not Alice holds Bob's test word, y, the projected F_p changes by more than a constant. Consequently, we invoke the same proof for F_p, $p > 1$ and obtain the same $2^{\Omega(d)}$ lower bound.

On the other hand, suppose that $p < 1$. We assume a code $C \subset \mathcal{B}(d, \epsilon d)$ with the property that any distinct $x, x' \in C$ have $|x \cap x'| \le cd$ for some small constant $c > \epsilon^2$ (see Lemma 3.2). Again, Alice holds a subset $T \subseteq C$ and inserts $\mathsf{star}(T)$ into the table for the problem A. Throughout this proof we use a binary alphabet so suppress the Q notation from $\mathsf{star}^Q(\cdot)$. Bob holds a test vector $y \in C$ and is tasked with determining whether or not Alice holds $y \in T$. We distinguish between the cases when Alice holds $y \in T$ or not as follows. Bob uses y to determine the query column set $S = \mathsf{supp}(y)$ and will compare against the returned frequency value from the algorithm.

Case 1: $y \notin T$. Consider some $y' \in C \setminus \{y\}$. Since y and y' are both codewords, they can have a 1 coincident in at most cd locations. So if Alice does not hold y then the codewords we need to consider are all binary words in the code which have at most cd 1s in common with y on S. We denote this collection of words by M, i.e., the set of binary strings of length d that have at most cd locations set to 1. There are r such vectors, where r is defined by:

$$r \triangleq \sum_{i=0}^{cd} \binom{d}{i} \le cd \cdot \binom{d}{cd} = O(d) 2^{\Theta(cd)}.$$

The total count of all strings generated by Alice's encoding is at most $2^{\epsilon d}|C|$: each string in C generates $2^{\epsilon d}$ subwords from the $\mathsf{star}(\cdot)$ operation. We now evaluate the ℓ_p-frequency of elements in the set M, denoted $F_p(M)$. For $p < 1$, the value $F_p(M)$ is maximized when every element of M has the same number of occurrences, $|C|2^{\epsilon d}/r$. As there are at most r members of M, we obtain $F_p(M) \le |C|^p 2^{\epsilon dp} r^{1-p}$. Recalling the bounds on $|C|$ and r, this is:

$$2^{cdp + \epsilon dp + \Theta((1-p)cd)} \cdot O(d^{1-p}). \tag{5}$$

We can now choose c to be a small enough constant so that (5) is at most $2^{(1-\alpha)\epsilon d}$ for a constant $\alpha > 0$ by Lemma A.2 in Appendix A.2.

Case 2: $y \in T$**.** Now consider the scenario when $y \in T$ so that Alice has inserted $\mathsf{star}(y)$ into the table A. Here, we can be sure that each of the $2^{\epsilon d}$ strings in $\mathsf{star}(y)$ appears at least once over the column set S, and so the F_p value is at least $2^{\epsilon d} 1^p = 2^{\epsilon d}$.

We observe that these two cases obtain the constant factor separation, as required. Then, Bob can use his test vector y and a query S with a constant factor approximation algorithm to the projected F_p-estimation problem and distinguish between the two cases of

Alice holding y or not. Thus, Bob can determine if $y \in T$ and consequently solve the Index problem, incurring the $\Omega(|C|) = 2^{\Omega_c(d)}$ lower bound for a c arbitrarily small. □

Remark 2. *For $p > 1$ we adopt the same instance as in Theorem 5.3 so the instance is of size $(|T| + 1)2^{\varepsilon d} \times d$. On the other hand, for $0 < p < 1$, only the words in $\text{star}^Q(T)$ are required so A has size $|T|2^{\varepsilon d} \times d$.*

5.4 ℓ_p-Sampling

In the projected ℓ_p-sampling problem, the goal is to sample a row in A^C proportional to the p-th power of its number of occurrences. One approach to the standard (non-projected) ℓ_p-sampling problem on a vector x is to subsample and find the ℓ_p-heavy hitters [14]. Consequently, if one can find ℓ_p-heavy hitters for a certain value of p, then one can perform ℓ_p-sampling in the same amount of space, up to polylogarithmic factors. Interestingly, for projected ℓ_p-sampling, this is not the case, and we show for every $p \neq 1$, there is a $2^{\Omega(d)}$ lower bound. This is despite the fact that we can estimate ℓ_p-frequencies efficiently for $0 < p < 1$, and hence find the heavy hitters (Section 5.1).

Theorem 5.5. *Fix a real number $p > 0$ with $p \neq 1$, and let $\varepsilon \in (0, 1/2)$. Let $S \subseteq [d]$ be a column query and i be a pattern observed on the projected data A^S. Any algorithm which returns a pattern i sampled from a distribution $(p_1, \ldots, p_n)$, where $p_i \in (1 \pm \varepsilon)\frac{f^p_{e(i)}}{\|f(A,S)\|^p_p} + \Delta$ together with a $(1 \pm \varepsilon')$-approximation to p_i, $\Delta = 1/\text{poly}(nd)$ and $\varepsilon' > 0$ is a sufficiently small constant, requires $2^{\Omega(d)}$ bits of space.*

Proof. **Case 1: $p > 1$.** The proof of Theorem 5.3 argues that the vector $\mathbf{0}_S$ is a constant factor ℓ_p-heavy hitter for any $p > 1$ if and only if Bob's test vector y is in Alice's input set T, via a reduction from Index. That is, we argue that there are constants $C_1 > C_2$ for which if $y \in T$, then $f^p_{e(\mathbf{0}_S)} \geq C_1 F_p$, while if $y \notin T$, then $f^p_{e(\mathbf{0}_S)} < C_2 F_p$. Consequently, given an ℓ_p-sampler with the guarantees as described in the theorem statement, then the (empirical) probability of sampling the item $\mathbf{0}_S$ should allow us to distinguish the two cases. This holds even tolerating the $(1 + \varepsilon')$-approximation in sampling rate, for a sufficiently small constant ε'. In particular, if $y \in T$, then we will indeed sample $\mathbf{0}_S$ with $\Omega(1)$ probability, which can be amplified by independent repetition; whereas, if $y \notin T$, we do not expect to sample $\mathbf{0}_S$ more than a handful of times. Consequently, for $p > 1$, an ℓ_p-sampler can be used to solve the ℓ_p-heavy hitters problem with arbitrarily large constant probability, and thus requires $2^{\Omega(d)}$ space.

Case 2: $0 < p < 1$. We now turn to $0 < p < 1$. In the proof of Theorem 5.4, a reduction from Index is described where Alice holds the set T and Bob the string y. Bob can generate the set $\text{star}(y)$ of size $2^{\varepsilon d}$ which is all possible binary strings supported on the column query S. From this, Bob constructs the set $M' = \left\{z \in \text{star}(y) : |\text{supp}(z)| \geq \frac{\varepsilon d}{2}\right\}$. We observe that if $y \in T$ then at least half of the strings in $\text{star}(y)$ are supported on at least $\varepsilon d/2$ coordinates which implies $|M'| \geq 2^{\varepsilon d - 1}$. The total F_p in this case can be bounded by a contribution of $|M'|1^p + 2^{\varepsilon d}$. The first term arises from the $|M'|$ strings in M' with a frequency of 1, while the second term is shown in Case 1 of Theorem 5.4. Since $|M'| \leq 2^{\varepsilon d}$, we have that $F_p \leq 2^{\varepsilon d + 1}$ in this case. Consequently, the correct probability of ℓ_p-sampling returning a string in M' is at least $\frac{1}{4}$ for the "ideal" case of $\varepsilon = 0, \Delta = 0$. Even allowing $\varepsilon < \frac{1}{2}$ and $\Delta = 1/\text{poly}(nd)$, this probability is at least $1/10$.

Otherwise, if $y \notin T$, we exploit that $y' \neq y$ can coincide in at most $cd = O(\varepsilon^2 d)$ coordinates and $|\text{supp}(z)| \geq \varepsilon d/2 > cd$ for any $z \in M'$. Hence, no $z \in M'$ can occur in $\text{star}(y')$ for another $y' \in C \setminus \{y\}$ on the column projection S. In this case, there should be zero probability of sampling a string in M' (neglecting the trivial additive probability Δ).

To summarize, in the case that $y \in T$, by querying the projection S then a constant fraction of the F_p-mass is on the set M', whereas when $y \notin T$, then there is zero F_p-mass on the set M'. Since Bob knows M', he can run an ℓ_p-sampler and check if the output is in the set M', and succeed with constant probability. It follows that Bob can solve the Index problem (amplifying success probability by independent repetitions if needed), and thus again the space required is $2^{\Omega(d)}$. □

Remark 3. *For $p > 1$ we again adopt the same instance as in Theorem 5.3 which has size $(|T| + 1)2^{\varepsilon d} \times d$. However, for $0 < p < 1$, we require the instance from Theorem 5.4 so A has size $|T|2^{\varepsilon d} \times d$.*

6 PROJECTED FREQUENCY ESTIMATION VIA SET ROUNDING

Although our lower bounds rule out the possibility of computing constant factor approximations to projected frequency problems in sub-exponential space, it is still possible to compute non-trivial approximations using exponential space but still better than naïvely enumerating all column subsets of $[d]$. We design a class of algorithms that proceed by keeping appropriate sketch data structures for a "net" of subsets. The net has the property that for any query $C \subset [d]$ there is a $C' \subset [d]$ stored in the net which is not too different from C. We can then answer the query on C using the summary data structure computed for columnset C'. To formalize this approach we need some further definitions, the first of which conceptualizes the notion of a net over subsets.

Definition 6.1 (α-net of subsets). Let $\mathcal{P}([d])$ denote the power set of $[d]$. Fix a parameter $\alpha \in (0, 1/2)$. An α-net of $\mathcal{P}([d])$ is the set $\mathcal{N} = \{U : |U| \leq d/2 - \alpha d \text{ or } |U| \geq d/2 + \alpha d\}$ which contains all subsets $U \in \mathcal{P}([d])$ whose size is at most $d/2 - \alpha d$ or at least $d/2 + \alpha d$.

Let $H(x) = -x \log_2(x) - (1 - x)\log_2(1 - x)$ denote the *binary entropy function.*

Lemma 6.2. *Let $\mathcal{N}$ be an α-net for $\mathcal{P}([d])$. Then $|\mathcal{N}| \leq 2^{H(1/2 - \alpha)d + 1}$.*

Proof. The total number of subsets whose size is at most $d/2 - \alpha d$ is $\sum_{i \leq \alpha d}\binom{d}{i}$ and $\sum_{i \leq \alpha d}\binom{d}{i} \leq 2^{H(1/2 - \alpha)d}$ [8, Theorem 3.1]. By symmetry we obtain the same bound for the number of subsets of size at least $d/2 + \alpha d$, yielding the claimed total. □

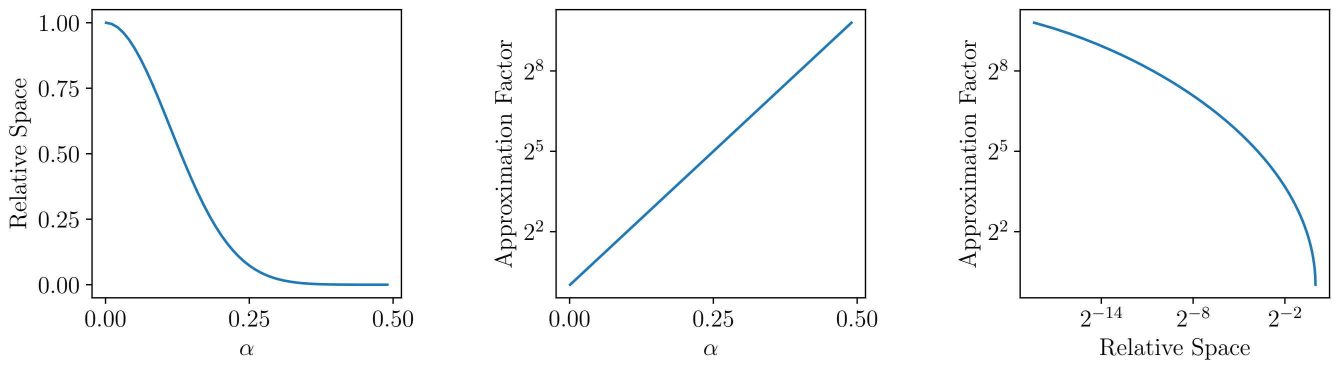

Figure 1: Space-approximation tradeoff for $d = 20$ as α is varied from 0 to $1/2$. Relative space is $2^{H(1/2-\alpha)d}/2^d$.

Algorithm 1: Projected frequency by query rounding

```
Input: Data $A \in \{0,1\}^{n\times d}$, parameter $\alpha \in (0,1/2)$,
       frequency estimation problem $P$, query $C$ revealed
       after $A$
1 Function ProjectedFreq($A, \alpha, C$):
2     Generate an $\alpha$-net $\mathcal{N}$
3     For every $U \in \mathcal{N}$ evaluate a $\beta$ approximate sketch to
        estimate $P(A, U)$
4     Given a projection query $C$ after observing $A$:
5     Obtain $C'$, an $\alpha$-neighbour to $C$ in $\mathcal{N}$
6     return $P(A, C')$ to $\beta$ relative error
```

6.1 From α-nets to Projections

Suppose that we are tasked with answering problem $P = P(A, C)$ on a projection query C. We know that if C is known ahead of time then we can encode the input data $A \in [Q]^{n\times d}$ on projection C as a standard stream over the alphabet $[Q]^{|C|}$. The use of α-nets allows us sketch some of the input and use this to approximately answer a query. For a standard streaming problem, we will say that an algorithm yields a *β-approximation* to the true solution z^* if the returned estimate $z \in [z^*/\beta, \beta z^*]$. A sketch obtaining such approximation guarantees will be referred to as a β approximate sketch. We additionally need the following notion of error due to the distortion incurred when answering queries on elements of the α-net rather than the given query.

Definition 6.3 (Rounding distortion). Let $P = P(A, C)$ be a projection query for the problem P on input $A \in [Q]^{n\times d}$ with projection C. Let $\mathcal{N} \subset \mathcal{P}([d])$ be an α-net. The *rounding distortion* $r(\alpha, P)$ is the worst-case determinstic error incurred by solving $P(A, C')$ rather than $P(A, C)$ for an α-neighbour $C' \in \mathcal{N}$ of C so that $P(A, C)/r(\alpha, P) \le P(A, C') \le r(\alpha, P)P(A, C)$.

Definition 6.3 is easiest to conceptualize for the F_0 problem when $A \in \{0,1\}^{n\times d}$. Specifically, $P = F_0$ and the task to solve is $P = F_0(A, C)$. For a given query C, with an α-neighbour C' in the net, the gap between the number of distinct items observed on C' at most doubles for each column in the set difference between C and C'. Since C' is an α-neighbour, we have $|C' \Delta C| \le \alpha d$ so the worst-case approximation factor in the number of distinct items observed over C' rather than C is $2^{\alpha d}$.

More generally, we can categorize the rounding distortion for other typical queries, as demonstrated in the following lemma. Note that if the query is contained in the α-net $\mathcal{N}$ then we will retain a sketch for that problem; hence the distortion is only incurred for queries not contained in the net.

Lemma 6.4. *Fix $\alpha \in (0, 1/2)$, suppose $A \in \{0,1\}^{n\times d}$ and $\mathcal{N}$ be an α-net. If C is a projection query for the following cases, the rounding distortion can be bounded as:*

(1) $P = F_0(A, C)$ then $r(\alpha, F_0) = 2^{\alpha d}$
(2) $P = F_p(A, C), p > 1$ then $r(\alpha, F_p) = 2^{\alpha d(p-1)}$
(3) $P = F_p(A, C), p < 1$ then $r(\alpha, F_p) = 2^{\alpha d(1-p)}$

Proof. Item (1) is an immediate consequence of the discussion above following Definition 6.3 so we focus on (2) and (3). Suppose $p \ge 1$. Let $f_C = f(A, C)$ denote the frequency vector associated to the projection query C over domain $[2^{|C|}]$. First, consider a single index $j \in [2^{|C|}]$ with $(f_C)_j = x$. Let C' be an α-neighbour for C in $\mathcal{N}$, and without loss of generality, assume that $|C| < |C'|$. The task is to estimate $\|f_C\|_p^p = x^p$ from $\|f_{C'}\|_p^p$, where $f_{C'} = f(A, C')$ is a frequency vector over the domain $[2^{|C'|}]$ which is a $|C' \setminus C|$ factor larger than the domain for f_C. However, observe that in $f_{C'}$, the value of x is spread across the at most $2^{\alpha d}$ entries that agree with j on columns C. The contribution to F_p from these entries is at most x^p (if the mass of x is mapped to a single entry). On the other hand, by Jensen's inequality, the contribution is at least $2^{\alpha d}(x/2^{\alpha d})^p = x^p/2^{\alpha d(p-1)}$. Hence, considering all entries j, we obtain $\|f_C\|_p^p/2^{\alpha d(p-1)} \le \|f_{C'}\|_p^p \le \|f_C\|_p^p$. In the case $|C| > |C'|$, essentially the same argument shows that $\|f_C\|_p^p \le \|f_{C'}\|_p^p \le \|f_C\|_p^p 2^{\alpha d(p-1)}$. Thus we obtain the rounding distortion of $2^{\alpha d(p-1)}$. For $p < 1$, we proceed as above, except by concavity, the ordering is reversed. □

Observe that the distortion reduces to 1 (no distortion) as we approach $p = 1$ from either side. This is intuitive, since the F_1 problem is simply to report the number of rows in the input, regardless of C, and so the problem becomes "easier" as we approach $p = 1$.

With these properties in hand, we can give a meta algorithm as described in Algorithm 1. In Theorem 6.5 we can fully characterize the accuracy-space tradeoff for Algorithm 1 as a function of α and d.

Theorem 6.5. *Let $A \in \{0,1\}^{n \times d}$ be the input data and $C \subseteq [d]$ be a projection query. Suppose $P = P(A, C)$ is the projected frequency problem, $\alpha \in (0, 1/2)$ and $r(\alpha, d)$ is the rounding distortion. With probability at least $1 - \delta$ a $\beta r(\alpha, d)$ approximation can be obtained by keeping $\tilde{O}(2^{H(1/2-\alpha)d})$ β-approximate sketches.*

Proof. Let $\mathcal{N}$ be a α-net for $\mathcal{P}([d])$ and for every $U \in \mathcal{N}$ generate a sketch with accuracy parameter ϵ for the problem P on the projection defined by $U \subseteq [d]$. Either the projection $C \in \mathcal{N}$, in which case we can report a β factor approximation, or $C \notin \mathcal{N}$ in which case we take an α-neighbour, $C' \in \mathcal{N}$ and return the estimate z for $P(A, C')$. The sketch ensures that the answer to $P(A, C')$ is obtained with accuracy β, which by the rounding distortion is a $\beta r(\alpha, d)$ approximation. To obtain this guarantee we build one sketch for every $U \in \mathcal{N}$, for a total of $O(2^{H(1/2-\alpha)d})$ sketches (via Lemma 6.2). By setting the failure probabilty for each sketch as $\delta = 1/2^{\alpha d}$ and then taking a union bound over the α-net we achieve probability at least $1 - \delta$. □

We remark that similar results are possible for the other functions considered, ℓ_p frequency estimation, ℓ_p heavy hitters and ℓ_p sampling. The key insight is that all these functions depend at their heart on the quantity $f_j/\|f\|_p$, the frequency of the item at location j divided by the ℓ_p norm. If we evaluate this quantity on a superset of columns, then both the numerator and denominator may shrink or grow, in the same ways as analyzed in Lemma 6.4, and hence their ratio is bounded by the same factor, up to a constant. Hence, we can also obtain (multiplicative) approximation algorithms for these problems with similar behavior.

Illustration of Bounds. First, observe that, irrespective of the problem P, the number of sketches needed is sublinear in 2^d. This is due to the fact that the entropy $H(1/2 - \alpha) < 1$ for $\alpha > 0$, so the size of the net $|\mathcal{N}| < 2^d$. For $0 \le p \le 2$, we have β-approximate sketches with $\beta = (1+\epsilon)$ whose size is $\tilde{O}(\varepsilon^{-2})$, which is constant for constant ϵ. For example, we obtain a $2^{\alpha d}$ approximation (ignoring small constant factors) for F_0 in space $O(2^{H(1/2-\alpha)}d)$, using for instance the $(1+\epsilon)$-approximate sketch from [11] which requires $O(\varepsilon^{-2} + \log n')$ bits for an input over domain $\{1, \ldots, n'\}$. Since $n' \le 2^d$, and setting $\epsilon = 1$, we obtain the approximation in space $O(d2^{H(1/2-\alpha)d})$. This is to be compared to the bounds in Section 4, where it is shown that (binary) instances of the projected F_0 problem require space $2^{\Omega(d)}$. These results show that the constant hidden by the $\Omega()$ notation is less than 1.

In Figure 1 we illustrate the general behavior of the bounds for $d = 20$. We plot the *relative space* by $2^{H(1/2-\alpha)}/2^d$ while varying α over $(0, 1/2)$ (plotted in the leftmost pane). This shows the space reduction in using the α-net approach compared to naiv̈ely storing all 2^d queries. The central pane shows how the approximation factor $2^{\alpha d}$ (on a log scale) varies with α. We plot the space-approximation tradeoff in the rightmost pane and the approximation factor is again plotted on a $\log_2$-scale. This plot suggests that if we reduce the space by a factor of 4 (i.e., permit relative space 2^{-2}) then the approximation factor is on the order of 10s. Meanwhile, if we use relative space 2^{-8}, then the approximation remains on the order of hundreds: this is a substantial saving as the number of summaries kept for the approximation is $2^{12} = 4096 \ll 2^{20} \approx 10^6$.

7 CONCLUDING REMARKS

We have introduced the topic of projected frequency estimation, with the aim of abstracting a range of problems involving computing functions over projected subspaces of data. Our main results show that these problems are generally hard, in terms of the space requirements: in most cases, we require space which is exponential in the dimensionality d of the input. However, interestingly, the exact dependence is not as simple as 2^d: we show that coarse approximations can be obtained whose cost is substantially sublinear in 2^d. Letting $N = 2^d$, our upper and lower bounds establish that the space complexity for a number of problems here is polynomial in N, though substantially sublinear. And, in a few special cases (ℓ_p frequency estimation for $p \le 1$), a sufficiently constant-sized sample suffices for accurate approximation of projected frequencies. It remains an intriguing open question to close the gaps between the upper and lower bounds, and to find the exact form of the polynomial dependence on N for these problems.

Acknowledgements. We thank S. Muthukrishnan and Jacques Dark for helpful discussions about this problem. The work of GC and CD was supported by European Research Council grant ERC-2014-CoG 647557. The work of DW was supported by NSF grant No. CCF-1815840, National Institute of Health grant 5R01HG 10798-2, and a Simons Investigator Award.

REFERENCES

[1] N. Alon, Y. Matias, and M. Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. *JCSS: Journal of Computer and System Sciences* 58 (1999), 137–147.

[2] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Val Tannen. 2016. Algorithms for Provisioning Queries and Analytics. In *International Conference on Database Theory*. 18:1–18:18.

[3] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P Woodruff. 2017. BPTree: An ℓ_2 heavy hitters algorithm sing constant memory. In *Proceedings of Principles of Database Systems*. ACM, 361–376.

[4] Vladimir Braverman, Elena Grigorescu, Harry Lang, David P. Woodruff, and Samson Zhou. 2018. Nearly Optimal Distinct Elements and Heavy Hitters on Sliding Windows. In *Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX/RANDOM 2018)*, Vol. 116. 7:1–7:22. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2018.7

[5] Vladimir Braverman, Robert Krauthgamer, and Lin F. Yang. 2018. Universal Streaming of Subset Norms. *CoRR* abs/1812.00241 (2018). arXiv:1812.00241 http://arxiv.org/abs/1812.00241

[6] Pern Hui Chia, Damien Desfontaines, Irippuge Milinda Perera, Daniel Simmons-Marengo, Chao Li, Wei-Yen Day, Qiushi Wang, and Miguel Guevara. 2019. KHyperLogLog: Estimating Reidentifiability and Joinability of Large Data at Scale. In *IEEE Symposium on Security and Privacy (SP)*. 867–881.

[7] Benjamin Doerr. 2020. Probabilistic tools for the analysis of randomized optimization heuristics. In *Theory of Evolutionary Computation*. Springer, 1–87.

[8] David Galvin. 2014. Three tutorial lectures on entropy and counting. *arXiv preprint arXiv:1406.7872* (2014).

[9] Rajesh Jayaram and David P. Woodruff. 2018. Perfect ℓ_p Sampling in a Data Stream. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 544–555.

[10] T. S. Jayram and D. P. Woodruff. 2009. The Data Stream Space Complexity of Cascaded Norms. In *IEEE Symposium on Foundations of Computer Science (FOCS)*. 765–774. https://doi.org/10.1109/FOCS.2009.82

[11] Daniel M Kane, Jelani Nelson, and David P Woodruff. 2010. An Optimal Algorithm for the Distinct Elements Problem. In *Proceedings of Principles of database systems*. ACM, 41–52.

[12] Ilan Kremer, Noam Nisan, and Dana Ron. 1999. On Randomized One-Round Communication Complexity. *Computational Complexity* 8, 1 (1999), 21–49.

[13] Branislav Kveton, S. Muthukrishnan, Hoa T. Vu, and Yikun Xian. 2018. Finding Subcube Heavy Hitters in Analytics Data Streams. In *Proceedings of the 2018 World Wide Web Conference*. 1705–1714. https://doi.org/10.1145/3178876.3186082

[14] Kasper Green Larsen, Jelani Nelson, Huy L. Nguyen, and Mikkel Thorup. 2016. Heavy Hitters via Cluster-Preserving Clustering. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*. 61–70.

[15] Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace Clustering for High Dimensional Data: a review. *SIGKDD Explorations* 6, 1 (2004), 90–105. https://doi.org/10.1145/1007730.1007731

[16] Sublinear.info. [n.d.]. Open Problem 94. https://sublinear.info/index.php?title=Open_Problems:94.

[17] Srikanta Tirthapura and David P. Woodruff. 2012. A General Method for Estimating Correlated Aggregates over a Data Stream. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*. 162–173.

[18] Hoa Vu. 2018. *Data Stream Algorithms for Large Graphs and High Dimensional Data*. Ph.D. Dissertation. U. Massachusetts at Amherst.

A OMITTED PROOFS

A.1 Omitted Proof for Section 5.1

Theorem A.1 (Restated Theorem 5.1). *Let $A \in \{0,1\}^{n \times d}$ be the input data and let $C \subseteq [d]$ be a given column query. For a given string $b \in \{0,1\}^C$, the absolute frequency of b, $f_{e(b)}$, can be estimated up to $\varepsilon \|f\|_1$ additive error using a uniform sample of size $O(\varepsilon^{-2}\log(1/\delta))$ with probability at least $1-\delta$.*

Proof. Let $T = \{i \in [n] : A_i^C = b\}$ be the set of indices on which the projection onto query set C is equal to the given pattern b. Sample t rows of A uniformly with replacement at a rate $q = t/n$. Let the (multi)-subset of rows obtained be denoted by B and the matrix formed from the rows of B be denoted $\hat{A}$. For every $i \in B$, define the indicator random variable X_i which is 1 if and only if the randomly sampled index i satisfies $A_i^C = b$, which occurs with probability $|T|/n$. Next, we define $\hat{T} = T \cap B$ so that $|\hat{T}| = \sum_{i=1}^t X_i$ and the estimator $Z = \frac{n}{t}|\hat{T}|$ has $\mathbb{E}(Z) = |T|$. Finally, apply an additive form of the Chernoff bound:

$$\begin{aligned}\mathbb{P}\left(|Z - \mathbb{E}(Z)| \geq \varepsilon n\right) &= \mathbb{P}\left(\left|\frac{n}{t}|\hat{T}| - |T|\right| \geq \varepsilon n\right) \\ &= \mathbb{P}\left(\left||\hat{T}| - \frac{t}{n}|T|\right| \geq \varepsilon t\right) \\ &\leq 2\exp\left(-\varepsilon^2 t\right).\end{aligned}$$

Setting $\delta = 2\exp\left(-\varepsilon^2 t\right)$ allows us to choose $t = O(\varepsilon^{-2}\log(1/\delta))$, which is independent of n and d. The final bound comes from observing that $\|f\|_1 = n$, $f_{e(b)} = |T|$ and $\hat{f}_{e(b)} = Z$. □

A.2 Omitted Proof for Section 5.3

A key step in the proof of Theorem 5.4 is that in Equation (5), the expression

$$2^{cdp+\epsilon dp+\Theta((1-p)cd)} \cdot O(d^{1-p})$$

can be bounded by a manageable power of two. We formalize this in Lemma A.2.

LEMMA A.2. *Under the same assumptions as in Theorem 5.4, there exists a small constant $c > 0$ which bounds Equation (5) by at most $2^{(1-\alpha)\epsilon d}$ for some $\alpha > 0$.*

PROOF. Here we use base-2 logarithms and let $0 < c < 1$ be a small constant which we need to bound. Also, let $0 < p < 1$ be a given constant. Observe that the $O(d^{1-p})$ term only contributes positively in the exponent term of (5) so we can ignore it from the calculation. Write $2^{\Theta(cd(1-p))} = 2^{cd\alpha(1-p)}$ for $\alpha > 0$. This follows from:

$$\binom{d}{cd} \leq \left(\frac{ed}{cd}\right)^{cd} \leq 2^{(2+\log\frac{1}{c})cd} \tag{6}$$

so let $\alpha = 2 + \log\frac{1}{c}$. For clarity, we proceed by using the trivial identity $1 - (1 - \nu) = \nu$ and show that $1 - \nu > 0$ for ν a function of c, p, d. We need to ensure:

$$cpd + \epsilon dp + \alpha cd(1-p) \leq (1-\alpha)\epsilon d. \tag{7}$$

This amounts to showing that:

$$\nu \triangleq cp/\epsilon + p + \alpha c(1-p)/\epsilon \leq (1-\alpha)$$

Now, $\nu = p(c/\epsilon + 1 - \alpha c/\epsilon) + \alpha c/\epsilon$ and we require $\nu < 1$. We may enforce the weaker property of $p(c/\epsilon + 1 - \alpha/\epsilon) < 1$ because $c > 0$ and for $c < 4$ we also have $\alpha > 0$ (inspection on Equation (6)) so $\alpha c/\epsilon > 0$, and so can be omitted. Solving for c we obtain $c(1-\alpha) < \epsilon(1/p - 1)$. Recalling the definition of α this becomes:

$$c(\log c - 1) < \epsilon(1/p - 1) \tag{8}$$

from which positivity on c yields $c \log c < \epsilon(1/p - 1)$. Hence, it is enough to use $c < \epsilon(1/p - 1)$. □