On Finding Frequent Elements in a Data Stream

Ravi Kumar[1] and Rina Panigrahy[2]

[1] Yahoo! Research, 701 First Ave, Sunnyvale, CA 94089, USA,
ravikumar@yahoo-inc.com,
[2] Microsoft Research, 1065 La Avenida, Mountain View, CA 94043, USA,
rina@microsoft.com

Abstract. We consider the problem of finding the most frequent elements in the data stream model; this problem has a linear lower bound in terms of the input length. In this paper we obtain sharper space lower bounds for this problem, not in terms of the length of the input as is traditionally done, but in terms of the quantitative properties (in this case, distribution of the element frequencies) of the input *per se*; this lower bound matches the best known upper bound for this problem. These bounds suggest the study of data stream algorithms through an instance-specific lens.

1 Introduction

The data stream model of computation has evolved into an indispensable tool to cope with processing massive data sets. In this model, the input is processed as and when it arrives by a computational device that has only a limited storage space. The complexity in this model is typically measured in terms of the space required by the algorithm; this is often quantified as a function of the length of the input (n) and the size of the domain from which the elements of the inputs are drawn (m). Data stream algorithms are deemed interesting, efficient, and usable only if their space requirement is sublinear in n; this is especially so since n is assumed to be extremely large. Efficient data stream algorithms are out of question for most problems if neither randomization nor approximation is permitted. With randomization, sublinear space data stream algorithms have been developed for approximating several statistical data analysis primitives, ranging from frequency moments and L_p differences to histograms. For an excellent account of the important results in the area, see the recent monograph by Muthukrishnan [1] and the survey by Babcock et al [2].

Motivation. Even though many problems are shown to have efficient data stream algorithms, it is provably impossible to obtain $o(n)$ space algorithms for many other interesting data processing problems, even with the allowances of approximation and randomization. This is a consequence of both the limitations of the data stream model itself and the inherent difficulty of the underlying problem.

An important problem that falls in this doomed class is the *MostFrequent* problem: given an input sequence $X = x_1, \ldots, x_n$, output the element(s) that occur the most in the sequence X. This is an important and recurring problem in databases, information retrieval, and web analysis; for instance, search engines would like to quickly distill the frequently-issued queries from their daily query logs. Unfortunately, Alon, Matias, and Szegedy [3] showed an $\Omega(n)$ lower bound for estimating the frequency of the largest item given an arbitrary data stream. An approximation version of this problem involves finding an element whose frequency is close that of the optimum; unfortunately even this version needs linear space in the worst case. Throughout the rest of the paper, *MostFrequent* will always refer to the approximation version.

Despite this seemingly insurmountable lower bound for the *MostFrequent* problem, a wide variety of heuristics that work very well in practice have been proposed; for example, see [4–7]. These heuristics are successful because the lower bound assumes a certain (near uniform) distribution of the frequencies of the elements in the stream, where as, in practice one can exploit the fact that the frequencies are power law or Zipfian. Charikar, Chen, and Colton [7] obtained an algorithm along with a performance guarantee for a relaxed version of the *MostFrequent* problem — given a parameter ϵ they can find an element with frequency at least $(1-\epsilon)$ times that of the most frequent element in space $\tilde{O}(F_2/F_\infty^2)$, where F_2 is the second frequency moment of the input and F_∞ is the frequency of the most frequent input element.[3] Although F_2/F_∞^2 may be linear in n in the worst case, for many distributions including the Zipfian or power law distributions encountered in practice, it is sublinear. A natural question arises: for such (and other) distributions in order to find the most frequent elements, do we need $\Omega(F_2/F_\infty^2)$ space or can we do better?

Our Contributions. For the *MostFrequent* problem, where we are given the frequency distribution of the elements in a stream and an adversary is allowed to choose the order in which the elements arrive, we show tight space bound (up to logarithmic factors) of $\tilde{\Theta}(F_2/F_\infty^2)$, where F_2 is the second frequency moment of the input and F_∞ is the frequency of the most frequent input element. The upper bound can be inferred from the algorithm of Charikar, Chen, and Colton [7]; see also [8]. We show a matching lower bound for multi-pass data stream algorithms that are even allowed to approximate the most frequent element.[4] The lower bound is established by information-theoretic arguments as in Bar-Yossef et al [9] and consists of two steps. The first step involves obtaining an information complexity lower bound for a functional version of set disjointness. The second step constructs separate instances of the set disjointness problem, carefully interleaves them, and uses the lower bound from the first step together with additional information-theoretic arguments to obtain the final lower bound.

[3] For a sequence $X = x_1, \ldots, x_n$ where each $x_i \in Y$, let $f_y(X)$ denote the number of times $y \in Y$ appears in the sequence. Then, $F_2(X) = \sum_{y \in Y} f_y^2(X)$ and $F_\infty(X) = \max_{y \in Y} f_y(X)$.

[4] Since $F_0 \geq F_2/F_\infty^2$, it is tempting to exploit this relationship towards a lower bound in terms of F_0 via a reduction to the indexing problem. However, this approach does not seem to lead to tight bounds even for one-pass algorithms and certainly does not say much for multi-pass algorithms.

In particular, our bounds show the precise behavior of the space requirements as a function of the second and the infinity frequency moments of the input. In particular, it shows that the algorithm in [7] for Zipf distributions is essentially optimal.

At a philosophical level, our results suggest that the worst-case lower bounds for many data stream problems are perhaps way too pessimistic and stronger and better bounds could be stated if one were to take the properties of inputs themselves into account and parametrize the complexity of the algorithms differently. To quote [1]:

> "...the goal would be to find workable approximations under suitable assumptions on input data distributions."

Our results makes a modest progress towards this goal. In fact, we propose the study of data stream algorithms through an *instance-specific* lens. Roughly speaking, the plan is to obtain optimal space bounds, both upper and lower, for data stream algorithms parametrized in terms of the *actual* properties of the input sequence than just its length, n. We call these the *instance-specific data stream bounds.*

Notation. Let $X = x_1, \ldots, x_n$ denote the input sequence of length n. We assume that the input sequence X arrives in an arbitrary order. We work with randomized approximation algorithms, i.e., algorithms that given an error parameter $\epsilon > 0$ and a confidence parameter $\delta > 0$, output a value that is an $(1 \pm \epsilon)$ multiplicative approximation to the correct value, with probability at least $1 - \delta$. We use $\tilde{O}, \tilde{\Omega}, \tilde{\Theta}$ to suppress factors that are logarithmic in n.

2 Main Result

First, we formally state the *MostFrequent* problem. Let $F_\infty = F_\infty(X)$ denote the frequency of the most frequent element in the input sequence X. Let $F_2 = F_2(X)$ denote the second frequency moment of the input sequence X.

Problem 1 (MostFrequent). In the *MostFrequent* problem, we are given the distribution of the frequencies of the elements, i.e., we are told how many elements have a certain frequency but not the identity of those elements. The objective is to find an element with frequency at least $(1 - \epsilon) \cdot F_\infty$, in one or more passes over the input.

2.1 Upper Bound

For sake of completeness, we briefly discuss the known algorithm for the *MostFrequent* problem. For simplicity of exposition, we stated it as a two-pass algorithm; it is possible to make it run in a single pass.

The algorithm maintains h hash tables of size b buckets each. In the first pass, each input element is hashed into the h tables using pairwise-independent hash functions, keeping a counter per bucket. In the second pass, input elements that hash to a large number buckets with a high counter value — above a threshold τ — are reported as frequent elements.

Here is a glimpse of the analysis. Focusing on one hash table, if there are b buckets, then the expected count in a bucket cell is n/b. The variance of this count can be computed to be at most F_2/b. But the expected count of the bucket with the most frequent element is at least $n/b + F_\infty(1-1/b)$. So, as long as the gap between the expectations $F_\infty(1-1/b)$ is at least $c \cdot \sqrt{F_2/b}$ for sufficiently chosen constant $c > 0$, we can set the threshold τ so that there is a constant gap in the probability of a count exceeding the threshold for the frequent element versus any other element. This gives a bound on the number of buckets to be $b > \Omega(F_2/F_\infty^2)$. By using several (constant) such tables and using the median count, the probability gap can be amplified arbitrarily.

Theorem 1 ([7]). *There is a randomized data stream algorithm for the Most-Frequent problem that uses space $\tilde{O}(F_2/F_\infty^2)$.*

2.2 Tools for the Lower Bound

We assume the reader is familiar with communication complexity and information theory. See the book by Kushilevitz and Nisan [10] for the former and the book by Cover and Thomas [11] for the latter.

In our usage, $X \perp Y$ will denote the random variables X and Y are independent. We use $H(\cdot)$ to denote the entropy and $I(\cdot;\cdot)$ to denote the mutual information between two random variables.[5] Some basic properties of the information-theoretic quantities can be found in [11].

First, we need the following simple lemma that relates the entropies of two random variables to the mutual information with two other random variables that are independent.

Lemma 1. *For random variables A, B, X, Y such that $X \perp Y$, we have*

$$H(A) + H(B) \geq I(AB; XY) + I(A; B) \geq I(AB; X) + I(A; Y) + I(B; Y).$$

Proof. The first inequality follows easily since

$$H(A) + H(B) \;=\; H(AB) + I(A; B) \;\geq\; I(AB; XY) + I(A; B).$$

To show the second inequality, we first show

$$\begin{aligned} I(AB; XY) &= H(XY) - H(XY \mid AB) \\ &= H(X) + H(Y) - H(XY \mid AB) \quad \text{(by } X \perp Y\text{)} \\ &\geq H(X) + H(Y) - H(X \mid AB) - H(Y \mid AB) \quad \text{(by subadditivity)} \\ &= I(AB; X) + I(AB; Y). \end{aligned} \tag{1}$$

[5] Let A, B be random variables. Then, the entropy $H(A) = -\sum_a \Pr[A = a] \log \Pr[A = a]$, the conditional entropy $H(A \mid B) = \sum_b H(A \mid B = b) \Pr[B = b]$ and the mutual information $I(A; B) = H(A) - H(A \mid B) = H(B) - H(B \mid A)$.

Next, we show that

$$
\begin{aligned}
I(B;Y \mid A) + I(A;B) &= H(B \mid A) - H(B \mid A, Y) + H(B) - H(B \mid A) \\
&= H(B) - H(B \mid A, Y) \\
&\geq H(B) - H(B \mid Y) \quad \text{(by dropping condition)} \\
&= I(B;Y). \qquad (2)
\end{aligned}
$$

Finally, the second inequality follows as

$$
\begin{aligned}
I(AB;XY) + I(A;B) &\geq I(AB;X) + I(AB;Y) + I(A;B) \quad \text{by (1)} \\
&= I(AB;X) + I(A;Y) + I(B;Y \mid A) + I(A;B) \\
&\geq I(AB;X) + I(A;Y) + I(B;Y). \quad \text{by (2)}
\end{aligned}
$$

Next, we generalize this to a collection of random variables defined on a tree.

Theorem 2. *Given random variables $U_0, \ldots, U_{2^k-1}$ and $k+1$ random variables $X_0, \ldots, X_k$ that are independent. Consider a binary tree of height k, with $U_0, \ldots, U_{2^k-1}$ at the leaves in that order and each node is the labeled by the concatenation of all the variables in leaves of the subtree rooted at that node. That is, the parents of the leaves have labels $U_0U_1, U_2U_3, U_4U_5, ..., U_{2^k-2}U_{2^k-1}$ in that order; the root has label $U_0U_1..U_{2^k-1}$; and in general the j-th node from leftat the i-th level (root is at level 0) is labeled by $V_{ij} = U_{j2^i}U_{j2^i+1} \cdots U_{(j+1)2^i-1}$ where $i \in \{0, \ldots, k\}$, $j \in j(k,i) = \{0, \ldots, 2^{(k-i)} - 1\}$.*

Then

$$\sum_{j=0}^{2^k-1} H(U_j) \geq \sum_{i \in [k], j \in j(k,i)} I(V_{ij}; X_i).$$

Note that the order in which the U's appear in V_{ij} is immaterial as the order does not affect the entropy and the mutual information. All that is used here is that each node is labeled with a concatenation of labels of its descendant leaves in any order.

Proof. We prove by induction on k. The base case for $k = 1$ is given by the Lemma 1. For notational simplicity, we first illustrate the proof for $k = 2$.

$$
\begin{aligned}
&H(U_0) + H(U_1) + H(U_2) + H(U_3) \\
&= H(U_0U_1) + H(U_2U_3) + I(U_0;U_1) + I(U_2;U_3) \\
&\geq I(U_0U_1U_2U_3;X_0) + I(U_0U_1;X_1X_2) + I(U_2U_3;X_1X_2) + I(U_0;U_1) + I(U_2;U_3) \\
&\qquad \text{(by Lemma 1)} \\
&\geq I(U_0U_1U_2U_3;X_0) + I(U_0U_1;X_1) + I(U_0;X_2) + I(U_1;X_2) + I(U_2U_3;X_1) \\
&\qquad + I(U_2;X_2) + I(U_3;X_2). \quad \text{(by Lemma 1 applied twice)}
\end{aligned}
$$

This completes the proof for $k = 2$. For general $k > 2$, we proceed by induction. Let X' be a sequence of k independent random variables such that $X'_i = X_i$ for

$i = 0, \ldots, k-1$ and $X'_{k-1} = X_{k-1}X_k$.

$$
\begin{aligned}
&\sum_{j=0}^{2^k-1} H(U_j) \\
&= \sum_{j'=0}^{2^{k-1}-1} H(U_{2j'}U_{2j'+1}) + \sum_{j'=0}^{k} I(U_{2j'};U_{2j'+1}) \\
&\geq \sum_{i\in[k-1],j\in j(k-1,i)} I(V_{ij};X'_i) + \sum_{j'=0}^{k} I(U_{2j'};U_{2j'+1}) \\
&\qquad \text{(by induction hypothesis and using } X'\text{)} \\
&= \sum_{i\in[k-2],j\in j(k-2,i)} I(V_{ij};X_i) + \sum_{j'=0}^{k} I(U_{2j'}U_{2j'+1};X_{k-1}X_k) + \sum_{j'=0}^{k} I(U_{2j'};U_{2j'+1}) \\
&\geq \sum_{i\in[k-2],j\in j(k-2,i)} I(V_{ij};X_i) + \sum_{j'=0}^{k} I(U_{2j'}U_{2j'+1};X_{k-1}) \\
&\qquad + \sum_{j'=0}^{k} I(U_{2j'};X_k) + I(U_{2j'+1};X_k) \quad \text{(by Lemma 1 applied } k \text{ times)} \\
&= \sum_{i\in[k],j\in j(k,i)} I(V_{ij};X_i).
\end{aligned}
$$

Note that the above theorem easily extends to a conditional version where all entropies and mutual information quantities are conditioned on some random variable. Such a conditional version will be used later.

We consider both multi-party communication protocols. A protocol specifies the rules of interaction between the players and a transcript $\Pi(X)$ contains the messages exchanged between the players upon the input X. If A is a random variable corresponding to a distribution on the inputs, then $I(A;\Pi(A))$ measures the mutual information between the input and the transcript; this is called *information cost*. The *information complexity* of a function is the minimum information cost of a protocol that correctly computes the function. Conditional version of these quantities are defined in an analogous manner; the reader is referred to [9] for more details.

2.3 The *IndexedAnd* Problem

We consider the following communication problem with t players, where each player has a bit vector. The objective is to find an index in the vector so that all t players have a '1' in that index.

Problem 2 (IndexedAnd). In the *IndexedAnd* problem, there are t players each with a binary vector of length n. Viewing the input to the players as a $n \times t$

binary matrix, the goal of the players is to output an index $i \in [n]$, if any, so that the i-th row in the matrix is all 1's; here a i-th row corresponds to the i-th bit of all players. If there is no such row with all 1's, then any output is valid.

Note that *IndexedAnd* is a functional version of the t-player set disjointness problem. Despite this close relationship, it is not clear if the communication lower bounds for set disjointness directly translate to the *IndexedAnd* problem.

We consider the following input instances for the problem.

(1) NO instance. In the inputs corresponding to the NO instance, each row has one 1 with probability 1/2 at a random column. As stated before, any output is considered valid if the input is a NO instance.

(2) YES instance. The inputs corresponding to the YES instance is the same as above except that exactly one row has all 1's; the correct answer in this case would be the index of that row.

A random NO instance can be generated as follows. Let $D = (D_1, \ldots, D_n)$ where D_i denotes the column number in row i that is set to 1 with probability 1/2; so each D_i takes a random value in the range $[t]$. Let $X \in \{0,1\}^{n \times t}$ be the random variable denoting the input and let $X_1, \ldots X_n$ denote the rows of X; so $X_{ij} = 1$ with probability 1/2 if $D_i = j$ and always 0 otherwise. Note that $X_1, \ldots, X_n$ are independent random variables, even conditioned on D.

We consider protocols that work correctly on the above YES and NO promise instances; note that the protocol is allowed to output any answer on a NO instance. Let $\mathrm{IC}_{IndexedAnd} = \min_{\Pi} I(X; \Pi(X) \mid D)$ be the conditional information complexity over all protocols that solve the above promise instances of *IndexedAnd*. Note that we use the above distribution corresponding to the NO instances to measure the information complexity. Let Π be the (randomized) protocol that achieves the minimum conditional information cost.

We also need the one-bit version of the *IndexedAnd* problem.

Problem 3 (And). In the *And* problem, there are t players with one bit each. The inputs are restricted to the cases where either all the t players have 1's or at most one player as a 1. The objective is to decide which case.

2.4 Lower Bound for the *IndexedAnd* Problem

We show the main theorem in this section.

Theorem 3. *There is a constant c so that for $n > ct$, $\mathrm{IC}_{IndexedAnd} \geq \tilde{\Omega}(n/t)$. (Alternately, we may say $\mathrm{IC}_{IndexedAnd} \geq \tilde{\Omega}(n/t - c)$.)*

To show this theorem, we first argue that there is a row i about which the protocol does not reveal much information. For simplicity let $\mathrm{IC} = \mathrm{IC}_{IndexedAnd}$ and let $\Pi = \Pi(X)$.

Lemma 2. *There is a row $i \in [n]$ such that $\Pr[i$ is output in NO instance$] \leq 2/n$ and $I(X_i; \Pi \mid D) \leq (2/n) \cdot \mathrm{IC}$.*

Proof. Since $X_1, \ldots, X_n$ are independent given D, it follows (à la [9]) that

$$\begin{aligned}
\mathrm{IC} &= I(X; \Pi \mid D) \\
&= H(X_1, \ldots, X_n \mid D) - H(X_1, \ldots, X_n \mid \Pi, D) \\
&\geq \left(\sum_{i=1}^{n} H(X_i \mid D)\right) - \left(\sum_{i=1}^{n} H(X_i \mid \Pi, D)\right) \\
&= \sum_{i=1}^{n} I(X_i; \Pi \mid D). \qquad (3)
\end{aligned}$$

Since the protocol is allowed to output anything on a NO instance, by averaging over the rows, there are at least $n/2$ rows so that for such row i, $\Pr[i \text{ is output in NO instance}] \leq 2/n$. Using (3) and averaging once more over the rows, there are at least $n/2$ rows so that for each such row i, we have $I(X_i; \Pi \mid D) \leq 2 \cdot \mathrm{IC}/n$. Putting these together, the proof is complete.

Let D_{-i} denote the entries of D except for the entry i. We now obtain a conditional version of Lemma 2.

Lemma 3. *There is a row $i \in [n]$ and values d_{-i} for D_{-i} so that* $\Pr[i \textit{ is output in NO case} \mid D_i, D_{-i} = d_{-i}] \leq 6/n$ *and* $I(X_i; \Pi \mid D_i, D_{-i} = d_{-i}) \leq 6 \cdot \mathrm{IC}/n$.

Proof. We use the i provided by Lemma 2. First, we have the guarantee $I(X_i; \Pi \mid D_i D_{-i}) \leq 2 \cdot \mathrm{IC}/n$. This is equivalent to $E_{d_{-i}}[I(X_i; \Pi \mid D_i, D_{-i} = d_{-i}] \leq 2 \cdot \mathrm{IC}/n$ over different values d_{-i} of D_{-i}. So by Markov's inequality, for at least 2/3-fraction of values d_{-i} of D_{-i}, we have $I(X_i; \Pi \mid D_i, D_{-i} = d_{-i}) \leq 6 \cdot \mathrm{IC}/n$. Similarly, for at least 2/3-fraction of values d_{-i} of D_{-i}, we have $\Pr[i \text{ is output in NO case} \mid D_i, D_{-i} = d_{-i}] \leq 6/n$. Together the two events must happen in at least 1/3-fraction of the cases completing the proof.

To prove theorem 3 we provide a reduction from the *And* problem. A lower bound on the information complexity of the one-bit *And* problem was proved in [9, 12].

Theorem 4 ([9, 12]). $\mathrm{IC}_{And} \geq \tilde{\Omega}(1/t)$.

Proof (Proof of Theorem 3). To complete the proof, we provide a reduction. Given an instance of the one-bit *And* problem and a protocol for *IndexedAnd*, we embed the one-bit *And* problem in row i that is given by Lemma 3. We use the values of d_{-i} to fill rows other than i. Note that once the values of d_{-i} are known, these rows can be filled without any communication among players, but by just using the players' private randomness.

The protocol for *IndexedAnd* is run on this input. Answer YES if output of this protocol equals i and NO otherwise. Note that in the NO instance of one-bit AND chosen randomly by setting a random position to 1 with probability 1/2, the probability of outputting YES is at most $6/n$ and in the YES instance it is 1. Out of the total $t + 1$ NO instances of the *And* problem each occurs in this random distribution with probability at least $1/(2t)$. So the probability of outputting YES when any of the $t+1$ NO instances is implanted on row i is at most $12t/n$. For $n > ct$, this probability is small and results in a protocol for the *And* problem. Therefore, $6 * \mathrm{IC}/n \geq \mathrm{IC}_{And}$ and thus $\mathrm{IC} \geq \Omega(n) * \mathrm{IC}_{And}$.

2.5 Lower Bound for the *MostFrequent* Problem

We now prove a lower bound on the space required for the *MostFrequent* problem. The set up is that we are given a vector $f = \{(n_1, k_1), (n_2, k_2), \ldots, (n_s, k_s), (1, t)\}$ where n_i elements occur with frequency k_i ($i \in [s]$) and the most frequent element occurs with frequency t that is the greater than all other frequencies k_i.

For the purposes of proving a lower bound we will make the following relaxations on the frequency vectors and the input instance. We assume that all the frequencies k_i and t are powers of 2; for such distributions any hardness on finding the most frequent element will directly mean a hardness on finding an approximate most frequent element to within factor 2.

Allowing Randomness in Frequency Counts. Next, instead of producing a stream with frequency vector exactly f, we will allow some randomness in the number of elements with the different frequencies — the number of elements with frequency k_i will be sharply concentrated around n_i. Precisely, frequency counts are obtained by drawing elements from a universe U that is a disjoint union $U_1 \cup U_2 \cup \cdots \cup U_s$ where $|U_i| = 2n_i$. Each element of U_i occurs independently and randomly in the stream either 0 or k_i times with probability 1/2. So the expected count of number of elements with frequency k_i is n_i; if n_i is large, this count is sharply concentrated around its n_i. Each element of the universe U occurs in the stream independently and randomly in this way, except that one element may occur t times.

If there is such an element in the stream with frequency t, then the algorithm is required to identify that element; if no such element occurs any output is allowed.

Theorem 5. *The space complexity for the above problem is* $\tilde{\Omega}(\sum_i n_i k_i^2/t^2)$.

We prove the lower bound by reduction from the *IndexedAnd* problem.

Reduction Idea. If all the frequencies k_i were the same, say k (i.e., $s = 1$), then the reduction is easy: given an instance of the *IndexedAnd* problem with $2n$ rows and t/k players we can construct a sub-stream for each player where the element is present or absent based on the bit in the input. Further each element is replicated k times. The final stream is the concatenation of the sub-streams for each player, where each element occurs k times and the element corresponding to the all 1's row occurs t times. Recall that *IndexedAnd* problem is hard even if we restrict to YES instances where one row as all 1's and each of the other rows have a 1 in a random column with probability 1/2 and all 0's with probability 1/2. So if a YES instance is translated into a stream, the count of the number of elements with frequency k is sharply concentrated around n. By Theorem 3, the total communication complexity is at least $\tilde{\Omega}(\frac{n}{(t/k)})$. So the average communication between successive players among the t/k players is $\tilde{\Omega}(\frac{n}{(t/k)^2}) = \tilde{\Omega}(nk^2/t^2)$

Reduction for Multiple Frequencies. When the number of frequencies s is more than 1 the reduction is more complicated. Instead of working with one instance of the *IndexedAnd* problem we will look at s instances each with t/k_i players. We will then interleave the sub-streams corresponding the different players over all the instances in such as way as to be able to add up their individual communication complexities.

Formally, we prove the lower bound by assuming a data stream algorithm to solve the *MostFrequent* problem and use it to obtain a communication protocol for *IndexedAnd*. Consider a collection of s random NO instances of the *IndexedAnd* problem, where the i-th instance has $2n_i$ rows and t/k_i players. We will show how a solution to *MostFrequent* can be used to solve a random YES instance of *IndexedAnd* implanted on any of these instances. Construct an instance of *MostFrequent* as follows.

By assumption $t = 2^{t'}$. Also assume without loss of generality that all the frequencies between 1 and t that are powers of 2 are present — this is easily achieved by setting the n_i corresponding to the absent ones to 0. So, $s = t', k_i = 2^{i-1}, i = 1, \ldots, s$.

Interleaving the Different Inputs. A block in the stream is constructed for each player; the j-th player in the i-th instance (for $i \in \{1, .., t'\}$ and $j \in \{1, .., 2^{t'-i+1}\}$) constructs a block b_{ij} by looking at its input column and including k_i copies of the element if a bit 1 is present in the corresponding row and none otherwise. To construct the stream we first arrange the blocks in a binary tree of height $t' + 1$ with a dummy root; the blocks b_{1j} corresponding to the first instance are at the leaves in order of increasing j; the blocks b_{ij} corresponding to the i-th instance are at height i; and the children of the root are two blocks $b_{t',1}$ and $b_{t',2}$ corresponding to the $t' - 1$-th instance. We construct a data stream from these blocks by laying them down in the depth-first search (preorder) traversal order;

Observe that if one of the $s = t'$ instances of the *IndexedAnd* problem is a YES instance and all the others are NO instances then there is an element in the stream that occurs t times that corresponds to the solution to the YES instance. Further the number of elements in the stream with frequency k_i is sharply concentrated around n_i. The essential idea is that a protocol for the *MostFrequent* problem can be used to solve a YES instance implanted into one of the s instances and random NO instances for the others.

Let Y_i denote a random NO input for the i-th instance. Let $Y = (Y_1, \ldots, Y_s)$ denote the combination of all these inputs. Let D denote the union of all the conditioning variables for all the instances. The first instance has t blocks. Let Π_r denote the output of the block b_{1r} in the streaming algorithm; by the output of a block we mean the content in memory after processing the block. We will lower bound $\sum_{r=1}^{t} H(\Pi_r \mid D)$.

Define $V_{1,1} = \Pi_1, \ldots, \Pi_t$. Define $V_{2,1} = \Pi_1 \Pi_3 \ldots \Pi_{t-1}$ and $V_{2,2} = \Pi_2 \Pi_4 \ldots \Pi_t$. Recursively define $V_{i,1}, V_{i,2}, ..., V_{i,2^i}$ where $V_{i,2j-1}$ and $V_{i,2j}$ are obtained from $V_{i-1,j}$ by taking all odd and even elements respectively (for example $V_{3,1} = \Pi_1 \Pi_5 \ldots \Pi_{t-3}, V_{3,2} = \Pi_3 \Pi_7 \ldots \Pi_{t-1}$).

Lemma 4. *$I(V_{i,j}; Y_j \mid D) \geq \mathrm{IC}_{n_i,t/k_i}$ where $\mathrm{IC}_{n_i,t/k_i}$ is the conditional information complexity of the IndexedAnd problem with n_i rows and t/k_i players.*

Proof. For simplicity let us prove first for $i = j = 1$. Assume that $I(V_{1,1}; Y_1 \mid D) < \mathrm{IC}_{n_1,t/k_1}$. Again let D_i denote the conditioning variables for the i-th instance and D_{-i} denote the conditioning variables for all the other instances. Then there is a value d_{-i} for D_{-i} so that $I(V_{1,1}; Y_1 \mid D_1, D_{-1} = d_{-1}) < \mathrm{IC}_{n_1,t/k_1}$. Then we have a protocol for the first instance of the *IndexedAnd* problem with improved conditional information complexity than the best possible which is a contradiction. Given an *IndexedAnd* instance with n_1 rows and t players, it can be implanted in the first instance of the reduction. The other instances can be generated using private random coin tosses based on d_{-i}.

Similarly for $V_{2,1} = \Pi_1 \Pi_3 \ldots \Pi_{t-1}$ and $V_{2,2} = \Pi_2 \Pi_4 \ldots \Pi_t$, observe that because of our specific ordering of blocks in the stream outputs $\Pi_1, \Pi_3, \ldots, \Pi_{t-1}$ partition the stream into $t/2$ parts and each block of the second instance resides in a different partition. So again $I(V_{2,1}; Y_2 \mid D) > \mathrm{IC}_{n_2,t/k_2}$ as otherwise we have a protocol for the second instance of the *IndexedAnd* problem with improved conditional information complexity. Similarly since the outputs $\Pi_2, \Pi_4, \ldots, \Pi_t$ also separate the $t/2$ blocks of the second instance into different partitions we have $I(V_{2,2}; Y_2 \mid D) > \mathrm{IC}_{n_2,t/k_2}$.

In this manner, the outputs in $V_{i,j}$ can be used to design a protocol for the i-th instance of the *IndexedAnd* problem giving $I(V_{i,j}; Y_i \mid D) > \mathrm{IC}_{n_i,t/k_i}$.

We now view the variables $V_{i,j}$ arranged in a binary tree of depth t' to invoke Theorem 2; $V_{i-1,j}$ is the j-th node at depth $i-1$ and has children $V_{i,2j-1}$ and $V_{i,2j}$. Each left (right) child has the even (odd) alternate variables of its parent. The leaves of this tree are $\Pi_1, \ldots, \Pi_t$ (not in that order). Further as required by Theorem 2, each node $V_{i,j}$ in the tree is a concatenation of the labels of the leaves of the subtree rooted at that node (note that although the order in which they are concatenated is different from one in the statement of Theorem 2; as noted earlier, this ordering does not make any difference in the entropy values). So Theorem 2 along with lemma 4 and Theorem 3 gives

$$\begin{aligned}\sum_{i=1}^{t} H(\Pi_i \mid D) &\geq \sum_{i=1}^{t'} \sum_{j=1}^{2^i} I(V_{i,j}; Y_i | D) \\ &\geq \sum_{i=1}^{t'} \sum_{j=1}^{2^i} \mathrm{IC}_{n_i,t/k_i}\end{aligned}$$

$$\geq \sum_{i=1}^{t'} 2k_i \tilde{\Omega}(n_i/(t/k_i - c)))$$
$$= \sum_{i=1}^{t'} \tilde{\Omega}(n_i k_i^2/t) - O(t).$$

By averaging, for some $i \in [t]$, $H(\Pi_i \mid D) \geq \tilde{\Omega}(\sum n_i k_i^2/t^2) - O(1)$. Recall that the space complexity of the *MostFrequent* problem is at least $\max_i H(\Pi_i \mid D)$. So ignoring the constant additive term, the space complexity is at least $\tilde{\Omega}(\sum n_i k_i^2/t^2)$ completing the proof of Theorem 5.

References

1. Muthukrishnan, S.: Data Streams: Algorithms and Applications. Now Publishers Inc. (2005)
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the 21st ACM Symposium on Principles of Databases Systems. (2002) 1–16
3. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. Journal of Computer and System Sciences **58**(1) (1999) 137–147
4. Gibbons, P., Matias, Y.: New sampling-based summary statistics for improving approximate query answers. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. (1998) 331–342
5. Gibbons, P., Matias, Y.: Synposis data structures for massive data sets. In: Proceedings of 10th Annual ACM-SIAM Symposium on Discrete Algorithms. (1999) 909–910
6. Fang, M., Shivakumar, N., Garcia-Molina, H., Motwani, R., Ullman, J.: Computing iceberg queries efficiently. In: Proceedings of 22nd International Conference on Very Large Data Bases. (1996) 307–317
7. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Proceedings of the 29th International Colloquium on Automata, Languages, and Programming. (2002) 693–703
8. Indyk, P., Woodruff, D.P.: Optimal approximations of the frequency moments of data sets. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing. (2005) 202–208
9. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. Journal of Computer and System Sciences **68**(4) (2004) 702–732
10. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press (1997)
11. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley & Sons, Inc. (1991)
12. Chakrabarti, A., Khot, S., Sun, X.: Near-optimal lower bounds on the multiparty communication complexity of set-disjointness. In: Proceedings of the 18th Annual IEEE Conference on Computational Complexity. (2003) 107–117