A Direct-Sum Theorem for Read-Once Branching Programs*

Anup Rao[1] and Makrand Sinha[2]

1 **Computer Science and Engineering, University of Washington, Seattle WA, USA**
anuprao@cs.washington.edu
2 **Computer Science and Engineering, University of Washington, Seattle WA, USA**
makrand@cs.washington.edu

Abstract

We study a direct-sum question for read-once branching programs. If $M(f)$ denotes the minimum average memory required to compute a function $f(x_1, x_2, \dots, x_n)$ how much memory is required to compute f on k independent inputs that arrive in parallel? We show that when the inputs are sampled independently from some domain $\mathcal{X}$ and $M(f) = \Omega(n)$, then computing the value of f on k streams requires average memory at least $\Omega\left(k \cdot \frac{M(f)}{n}\right)$.

Our results are obtained by defining new ways to measure the information complexity of read-once branching programs. We define two such measures: the *transitional* and *cumulative* information content. We prove that any read-once branching program with transitional information content $\mathtt{I}$ can be simulated using average memory $\mathcal{O}(n(\mathtt{I}+1))$. On the other hand, if every read-once branching program with cumulative information content $\mathtt{I}$ can be simulated with average memory $\mathcal{O}(\mathtt{I}+1)$, then computing f on k inputs requires average memory at least $\Omega(k \cdot (M(f) - 1))$.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation

Keywords and phrases Direct-sum, Information complexity, Streaming Algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.44

1 Introduction

In this paper we investigate direct-sum questions for read-once branching programs (equivalently, streaming algorithms). Recall that an input to a read-once branching program is a sequence of n updates $x_1, \dots, x_n$ arriving sequentially in time, and the branching program at the end must compute a function $f(x_1, x_2, \dots, x_n)$. The complexity measure of interest is the amount of memory that is needed to carry out the computation. Here the memory used by the program at time t is the logarithm of the number of potential states that the program can be in after reading the inputs $x_1, \dots, x_t$.

We are interested in how the complexity of a problem changes when the branching program must process k independent inputs that arrive in parallel. The program now gets k inputs $x^1 = x^1_1, \dots, x^1_n, x^2 = x^2_1, \dots, x^2_n, \dots, x^k = x^k_1, \dots, x^k_n$, where the inputs $x^1_t, \dots, x^k_t$ arrive simultaneously in the t'th time-step. Obviously one can process each of the inputs independently, giving a branching program that uses k times as much memory. The central

* Supported by the National Science Foundation under agreement CCF-1016565, an NSF Career award, and by the Binational Science Foundation under agreement 2010089.

question that we investigate in this paper is: are there interesting functions f for which the best branching program that computes f on k independent inputs does *not* operate independently on each input? This question is dual to another interesting question: When can we effectively reduce the memory of a branching program without compromising its accuracy?

Viewing these read-once branching programs as streaming algorithms, these questions also make a lot of sense in the context of the most common applications for streaming algorithms like internet traffic analysis or data from multiple satellites. They also make sense from a theoretical perspective: they help to identify exactly what makes some streaming tasks hard and others easy.

The extensive literature on branching programs is mostly concerned with understanding the maximum number of bits of memory used by the branching program throughout its run. One can imagine pathological cases one can effectively process k inputs at the same cost as processing a single input using this measure of complexity. Suppose there is a uniformly random block of n/k^3 consecutive updates that contains information in the input, and all other updates are set to 0. Then without loss of generality, the best (read-once) branching program uses almost no memory for most of the time, and some memory to process the block of important inputs. When the program processes k parallel inputs, it is very likely that the k informative blocks will not overlap in time, and so the maximum memory usage remains unchanged. Thus, if we are only aiming for a read-once branching program that succeeds with high probability over this distribution of inputs, one need not increase the memory at all!

However, we see that the *average memory* usage per unit time-step does increase by a factor of k in this last example. The average memory is defined to be the number of bits of memory used on an average time-step. Arguably from the streaming viewpoint, the average memory is what we care about when considering practical applications of streaming algorithms. Another appealing reason to consider average memory as a complexity measure is that some known streaming lower bounds actually yield lower bounds on the average memory. For example, the lower bound proofs for approximating the frequency moments [1, 4, 10, 22] and for approximating the length of the longest increasing subsequence [19] can be easily adapted to give matching lower bounds for average memory. In the rest of this work we focus on the average memory used by the branching program.

Note that it is standard for analyzing branching programs to count the number of states in each layer, but since we will be working with entropy it will be more convenient for us to talk about the memory required to store each layer. As such to present our results we adopt the point of view of inputs as streams and a branching program as a streaming algorithm.

1.1 Related Work

The interest in the field of streaming algorithms was renewed by the seminal paper of Alon, Matias and Szegedy [1] who gave algorithms for approximating lower frequency moments and also showed that lower bounds in the multi-party number-in-hand communication model implied memory lower bounds for streaming algorithms approximating the higher frequency moments. Since then, lower bounds in communication complexity (and more recently in information complexity) have found applications in proving memory lower bounds in the streaming model (see [1, 4, 10, 32, 19, 23, 22, 28] for some of them).

Questions analogous to the ones we study here have been studied in the setting of two-party communication complexity and information complexity [5, 9, 6]. It was shown in [21] that there are communication tasks that can be solved much more efficiently in parallel than by naively solving each one independently.

Combining these results about parallelizing communication with known methods for proving lower bounds on streaming algorithms gives several interesting worst-case memory lower bounds for computing natural functions on k parallel streams. To give an example, it is known that computing $(1+\varepsilon)$ approximation of the p^{th} frequency moment for $p \neq 1$ requires worst-case memory $\Omega(1/\varepsilon^2)$ [32, 22]. Combining this with the results of [9] one can show that computing $(1+\varepsilon)$ approximation of the frequency moment on k streams in parallel requires $\Omega(k/\varepsilon^2)$ memory in the worst-case. We do not give the proof here, since it is relatively straightforward.

A related model is that of *dynamic distributed functional monitoring* introduced by Cormode, Muthukrishnan and Yi [15] where there are multiple sites receiving data streams and communicating with a central coordinator who wants to maintain a function of all the input streams. Recent progress has been made in understanding the communication complexity of various tasks in this model [15, 33, 34]. Variants of this model have been studied extensively in relation to databases and distributed computing (see [13, 14, 31, 30, 12, 16, 2, 27, 26, 3] for some of the applications). Another closely related model is the multi-party *private message passing* model introduced in [18]. Any lower bound proved in the message passing model implies a lower bound in the streaming model. Many works have studied this model and its variants (see [23, 20, 29, 7, 11, 25] for some of them). These works do not appear to have any connection to the questions we study here.

2 Our Results

Our results are proved in the setting of average-case complexity: we assume that there is a known distribution on inputs, and consider the performance of algorithms with respect to that distribution. Let $\mathcal{A}$ be a randomized streaming algorithm which receives an input stream $X_1, \dots, X_n$ sampled from a distribution $p(x_1, \dots, x_n)$. Throughout this paper we will only consider the case when p is a product distribution except in Section 4.1, where we discuss the issues that arise when considering non-product input distributions.

Let $M_1, \dots, M_n$ denote the contents of the memory of the algorithm at each of the time-steps. Let $|M_t|$ denote the number of bits used to store M_t. The average memory used by the algorithm is $(1/n)\sum_{t=1}^{n} |M_t|$. Let $M(f)$ denote the minimum average memory required to compute a function f with probability $2/3$ when the inputs are sampled according to p.

Let $p^k(x)$ denote the product distribution on k independent streams, each identically distributed as $p(x)$, where the resulting streams arrive synchronously in parallel. Thus at time t the input is the t^{th} element of all the k streams. Write f^k to denote the function that computes f on each of the k streams. Then we prove,

▶ **Theorem 2.1.**

$$M(f^k) = \Omega\left(k\left(\frac{M(f)}{n} - 1\right)\right).$$

Theorem 2.1 is proved by a reduction that compresses streaming algorithms with regards to its information complexity. There are several reasonable measures of information complexity for streaming algorithms. Here we define two such information complexity measures. We use Shannon's notion of mutual information, which is defined in the preliminaries (Section 3).

The *transitional information content* captures the average amount of information that the algorithm learns about the next input conditioned on its current state.

▶ **Definition 2.2** (Transitional Information). $\mathsf{IC}^{tr}(\mathcal{A}) = \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t | M_{t-1})$.

The *cumulative information content* measures the average amount of information that the streaming algorithm remembers about the inputs seen so far.

▶ **Definition 2.3** (Cumulative Information). $\mathsf{IC}^{cum}(\mathcal{A}) = \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_1 \ldots X_t)$.

Note that both the transitional and the cumulative information content for an algorithm are bounded by the average memory used by the algorithm. We prove that algorithms with low transitional information can be efficiently simulated:

▶ **Theorem 2.4.** *Every streaming algorithm with transitional information content* $\mathtt{I}$ *can be simulated with average memory* $\mathcal{O}(n\mathtt{I} + n)$.

The above theorem is tight as the following example shows. Let the input x be sampled from the uniform distribution on $\{0,1\}^n$ (*i.e.* each update x_i for $i \in [n]$ is a bit). Consider the streaming algorithm $\mathcal{A}$ which remembers all the updates seen so far and outputs $x_1, \ldots, x_n$ at the end. The average memory used by the algorithm is $\Omega(n)$ while the transitional information content of this algorithm is 1. In this case the compression algorithm given by the above theorem would simulate $\mathcal{A}$ with average memory $O(n)$ which is the best one could hope for.

Finally, we show that if algorithms with low cumulative information can be simulated, then one can obtain no savings when parallelizing streaming algorithms:

▶ **Theorem 2.5.** *If every algorithm with cumulative information* $\mathtt{I}$ *can be simulated using average memory* $\mathcal{O}(\mathtt{I})$*, then* $M(f^k) = \Omega\left(k \cdot (M(f) - 1)\right)$.

In Section 5, we discuss more about the possibility of compressing algorithms with low cumulative information content.

3 Preliminaries

Throughout this report, the base of all logarithms is 2. Random variables will be denoted by capital letters and the values that they attain will be denoted by lower-case letters. Given $x = x_1, \ldots, x_n$, we write $x_{\leq i}$ to denote the sequence $x_1, x_2, \ldots, x_i$. We define $x_{<i}, x_{>i}$ and $x_{\geq i}$ similarly. We write x_{-i} to denote $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$.

We use $p(x)$ to denote both the distribution on the variable x and the probability $\mathbb{P}_p[X = x]$, the distinction will be clear from context. For any joint random variables X and Y, we will write $X|Y = y$ to denote the random variable X conditioned on the event $Y = y$ and use $p(x|y)$ to denote the distribution of $X|Y = y$ as well as the probability $\mathbb{P}_p[X = x|Y = y]$.

We denote by $p^k(x)$ the product distribution sampling k independent copies of x according to p. Given a joint distribution $p(x, y, z)$, we write $p(x, y)$ to denote the marginal distribution (or probability according to the marginal distribution) on the variables x and y. We often write $p(xy)$ instead of $p(x, y)$ to make the notation more concise. When X, Y are random variables, XY denotes the random variable that is the concatenation of X and Y.

Let X, W, M be random variables distributed according to $p(x, w, m)$. We say that they form a Markov chain iff $p(x, w, m) = p(w) \cdot p(x|w) \cdot p(m|w)$ and we denote this by $X - W - M$. In some cases we will have Markov chains where W determines M ($p(m|w)$ is a point distribution). To emphasize this we will write this Markov chain as $X - W \to M$. For brevity we will write $X|R - W|R - M|R$ to assert that $p(xwm|r)$ is a Markov chain for every r.

3.1 Information Theory Basics

Here we collect some standard facts from information theory. For more details, we refer the reader to the textbook [17]. For a discrete random variable X with probability distribution $p(x)$, the entropy of X is defined as

$$H(X) = \mathbb{E}_{p(x)} \left[\log \frac{1}{p(x)} \right].$$

For any two random variables X and Y with the joint distribution $p(x, y)$, the entropy of X conditioned on Y is defined as $H(X|Y) = \mathbb{E}_{p(y)}[H(X|Y = y)]$. The conditional entropy $H(X|Y)$ is at most $H(X)$ where the equality holds if and only if X and Y are independent.

The mutual information between X and Y is defined as $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$. Similarly, the conditional mutual information $I(X;Y|Z)$ is defined to be $H(X|Z) - H(X|YZ)$. If X and Y are independent then $I(X;Y) = 0$. Moreover, $0 \leq I(X;Y) \leq \min\{H(X), H(Y)\}$. A standard fact about mutual information is the chain rule: For jointly distributed random variables $X_1, \ldots, X_n$, Y and Z,

$$I(X_1, \ldots, X_n; Y|Z) = \sum_{i=1}^{n} I(X_i; Y|X_{<i}Z).$$

▶ **Lemma 3.1.** *If Y and Z are independent, $I(X;Y) \leq I(X;Y|Z)$.*

Proof. We repeatedly use the chain rule:

$$I(X;Y) \leq I(X;Y) + I(Z;Y|X) = I(XZ;Y) = I(Z;Y) + I(X;Y|Z) = I(X;Y|Z).$$ ◀

▶ **Proposition 3.2** (Data Processing Inequality)**.** *Let X, W and M be random variables such that $X - W - M$, then $I(X;M) \leq I(X;W)$.*

▶ **Proposition 3.3.** *Let X, Y, Z and W be random variables such that $XY - Z - W$, then $I(X;Y|ZW) = I(X;Y|Z)$.*

Proof. Using the chain rule we expand $I(XW;Y|Z)$ in two different ways:

$$I(W;Y|Z) + I(X;Y|ZW) = I(XW;Y|Z) = I(X;Y|Z) + I(W;Y|XZ).$$

The terms $I(W;Y|Z)$ and $I(W;Y|XZ)$ are 0 since $XY - Z - W$. ◀

The next proposition says that for any discrete random variable X there is a prefix-free encoding with average length at most $H(X) + 1$.

▶ **Proposition 3.4** (Huffman Encoding)**.** *Let X and Y be random variables where X is discrete. Then, there exists a prefix-free encoding $\ell : \mathsf{supp}(X) \to \{0,1\}^*$ satisfying $\mathbb{E}_{xy}[|\ell(x)| \mid Y = y] \leq H(X|Y) + 1$.*

3.2 Common Information and Error-free Sampling

Wyner [35] defined the quantity *common information* between X and M as

$$\mathbb{C}(X;M) = \inf_{X-W-M} I(XM;W),$$

where the infimum is taken over all jointly distributed W such that, $X - W - M$ and W is supported over a finite set. Wyner showed that the above infimum is always achieved. By the data-processing inequality applied to the Markov chain $X - W - M$ it is easily seen that $\mathbb{C}(X;M) \geq I(X;M)$.

It turns out that the gap between $\mathbb{C}(X;M)$ and $I(X;M)$ can be very large. There are known examples of random variables X and M where $\mathbb{C}(X;M) = \omega(I(X;M))$. We include one simple example in Appendix A. Another example is described in the work of Harsha et al. [24], who also proved a related upper bound. They showed that there always exist C and S, where S is independent of X, $X - CS \to M$ and $H(C) \approx I(X;M)$. The random variable S in their work depends on the distribution of M. Braverman and Garg [8] showed a similar result that we quote and use in this work:

▶ **Lemma 3.5** ([8]). *Let $p(xm)$ be an arbitrary discrete probability distribution, with finite support. Let S be an infinite list of uniform samples from $\mathsf{supp}(M) \times [0,1]$, independent of XM. Then there exists a random variable C such that $X - CS \to M$ and $H(C|S) \leq I(X;M) + \log(I(X;M)+1) + \mathcal{O}(1)$.*

3.3 Streaming Algorithms

Without loss of generality, we associate the values stored by the algorithm with a non-negative integer. Assuming that the inputs to the algorithm come from the domain $\mathcal{X}$, a streaming algorithm defines a function $\mathcal{A} : [n] \times \mathbb{N} \times \mathcal{X} \to \mathbb{N}$. At time $t-1$, let the memory state of the algorithm be m_{t-1} (we define $m_0 := 1$). On seeing the input x_t at time t, the algorithm computes the t^{th} memory state $m_t := \mathcal{A}(t, m_{t-1}, x_t)$. The output of the algorithm is m_n. Randomized streaming algorithms toss *independent* random coins r_t at each time-step t and sample the memory state at time t as follows: $m_t := \mathcal{A}(t, m_{t-1}, r_t, x_t)$.

The following is obvious from the definition:

▶ **Proposition 3.6** (Markov Chain Property). *If $m_1, \ldots, m_n$ denote the memory of a (possibly randomized) streaming algorithm, then for each $t \in [n]$, $X_{\leq n} M_{<t} - X_t M_{t-1} - M_t$.*

The last proposition also implies the following.

▶ **Proposition 3.7.** *For a randomized streaming algorithm, the following holds,*

$$I(M_{\leq n}; X_{\leq n}) = I(M_1; X_1) + I(M_2; X_2|M_1) + \cdots + I(M_n; X_n|M_{n-1}).$$

Proof. Applying the chain-rule, we get

$$I(M_{\leq n}; X_{\leq n}) = \sum_{t=1}^{n} I(M_t; X_{\leq n}|M_{<t}) \leq \sum_{t=1}^{n} I(M_t; X_t X_{\leq n} M_{<t-1}|M_{t-1}).$$

The second inequality follows since $I(M_t; X_t X_{\leq n} M_{<t-1}|M_{t-1}) = I(M_t; M_{<t-1}|M_{t-1}) + I(M_t; X_{\leq n}|M_{<t}) + I(M_t; X_t|M_{<t} X_{\leq n})$ and mutual information is a non-negative quantity.

Applying the chain rule one more time, we have

$$\begin{aligned} I(M_{\leq n}; X_{\leq n}) &\leq \sum_{t=1}^{n} I(M_t; X_t X_{\leq n} M_{<t-1}|M_{t-1}) \\ &= \sum_{t=1}^{n} I(M_t; X_t|M_{t-1}) + \sum_{t=1}^{n} I(M_t; X_{\leq n} M_{<t-1}|X_t M_{t-1}). \end{aligned}$$

Proposition 3.6 implies that $X_{\leq n} M_{<t} - X_t M_{t-1} - M_t$ for every $t \in [n]$ and hence the second term on the right hand side is zero. ◀

The following proposition states that both the transitional and cumulative information content are upper bounded by the average memory.

▶ **Proposition 3.8.** *For a randomized streaming algorithm $\mathcal{A}$ with average memory M,*

$$\max\{\mathsf{IC}^{tr}(\mathcal{A}), \mathsf{IC}^{cum}(\mathcal{A})\} \leq M\,.$$

▶ **Definition 3.9** (Simulation)**.** We say that a streaming algorithm $\mathcal{A}_1$ simulates another algorithm $\mathcal{A}_2$ if for every input $x_1, \ldots, x_n$, the distribution on outputs is exactly the same in both algorithms.

In general it even makes sense to allow errors during simulation. Our simulations have no error, so we define simulation using the strong definition given above.

4 Compression and Direct Sums for Streaming Computation

The following is a natural strategy to prove our direct-sum theorem: given an algorithm that computes f^k correctly with probability $2/3$ on all the streams and uses average memory M, first show that there is some stream "with respect to" which the information content is M/k. Then derive a randomized streaming algorithm that computes f and has information content at most M/k as follows: embed the input stream at the location j about which the memory has small information and simulate the behavior of the algorithm on this stream by generating the other streams randomly, or to say alternately, sample from the distribution $p(m_n|X^{(j)} = x)$. The resulting algorithm would have information content at most M/k but would still use M bits of average memory. The last step would then be to give a way to simulate a streaming algorithm that has information content $\mathtt{I}$ with a streaming algorithm that uses average memory approximately $\mathtt{I}$.

For product distributions, we can show that if there exists an algorithm for computing k copies of f with memory M, then there is a randomized algorithm for computing a single copy of f with transitional and cumulative information content at most M/k. To prove our direct-sum result, we are able to show that algorithms with transitional information content I can be simulated with $\mathcal{O}(n\mathtt{I}+n)$ average memory which as discussed before is best possible. To give an optimal direct-sum result, one could still hope that streaming algorithms with cumulative information content I can be simulated with $\mathcal{O}(I)$ average memory. We discuss more about this possibility in Section 5.

4.1 Non-product Distributions and Correlated Randomness

Before we begin the proof of our compression and direct-sum results, we briefly discuss the difficulty that arises in dealing with non-product distributions. For proving a direct-sum result for non-product distributions using the above strategy, the natural way of using an algorithm that computes k copies of f to compute a single copy of f, is to embed our input stream at position j and generate other streams as randomness so that we can run the algorithm for k copies. The algorithm we get for computing f in this way uses randomness that is correlated across various time-steps if the input stream distribution is non-product.

Transitional information content is not a useful information measure for compressing such algorithms as the following example shows. We give an example of a function which require $\Omega(1)$ average memory, but can be computed by an algorithm that uses correlated randomness and has transitional information content $1/n$. Let $f(x) = \sum_{t=1}^{n} x_t \mod 2$. Consider the following algorithm that takes as input a random input stream x (each update x_t is a bit) and computes $f(x)$. The algorithm at time t uses randomness r_t where $r_1, \ldots, r_t$ are correlated so that they satisfy $\sum_{t=1}^{n} r_t = 0 \mod 2$. At time t, the algorithm stores in its memory $\sum_{i=1}^{t}(x_t + r_t) \mod 2$ and at time $t = n$ outputs the last value stored in memory. Since $\sum_{t=1}^{n} r_t = 0 \mod 2$, the algorithm outputs $f(x)$. This algorithm has transitional information content $1/n$, but one can not hope to compute the parity of an n bit string without using $\Omega(1)$ bits of average memory.

4.2 Compressing Streaming Algorithms

In this section we show how algorithms with small transitional information content can be simulated with small average memory.

▶ **Theorem 4.1** (Restated). *Let $\mathcal{A}$ be a randomized streaming algorithm with $\mathsf{IC}^{tr}(\mathcal{A}) = \mathtt{I}$. Then there exists a randomized streaming algorithm $\mathcal{A}_{tr}$ with average memory $\mathcal{O}(n\mathtt{I} + n)$ that simulates $\mathcal{A}$.*

Let $m_1, \ldots, m_n$ denote the memory states of the algorithm $\mathcal{A}$. Recall that Lemma 3.6 implies that for each $t \in [n]$, $X_{\leq n} M_{<t} - X_t M_{t-1} - M_t$. Hence, to prove Theorem 4.1, it suffices to sample from $p(m_t|x_t, m_{t-1})$ if m_{t-1} has been sampled correctly. The compression algorithm will toss random coins to sample an infinite list s_t of samples from $\mathsf{supp}(M_t) \times [0,1]$ and then sample C_t (whose existence is guaranteed by Lemma 3.5) satisfying

$$X_t - C_t S_t | M_{t-1} \to M_t | M_{t-1}, \tag{4.1}$$

$$H(C_t | S_t M_{t-1}) = I(M_t; X_t | M_{t-1}) + \log(I(M_t; X_t | M_{t-1}) + 1) + \mathcal{O}(1). \tag{4.2}$$

The value of m_t determined by the sample c_t is distributed according to the distribution $\mathsf{p}(m_t|x_t, m_{t-1})$.

The algorithm will store the Huffman encoding (Proposition 3.4) of C_t conditioned on S_t and M_{t-1}. This encoding determines C_t given S_t and M_{t-1}, both of which are known to the algorithm at this time.

Randomized Streaming Algorithm $\mathcal{A}_{tr}$

Input : Stream $x \sim p(x)$
Randomness: $s_1, \ldots, s_n$ where s_i is an infinite sequence of uniform samples from $\mathsf{supp}(M_i) \times [0,1]$.

`// At time` t: `the content of the memory are some encodings of` $c_{<t}$, `where` c_i `determines` m_i `given` s_i `and` m_{i-1}.

1. Let m_{t-1} be determined by c_{t-1} and s_{t-1}. On input x_t, sample c_t from the Markov chain in (4.1);
2. Append the Huffman encoding of c_t conditioned on s_t and m_{t-1} to the previous memory contents;

Note that the algorithm needs to store the encodings of all the previous $c_{\leq t}$ at time t since in order to determine m_t uniquely, the value of m_{t-1} needs to be known which depends on the previous memory contents.

The following proposition is straightforward from (4.1).

▶ **Proposition 4.2.** *The algorithm $\mathcal{A}_{tr}$ simulates $\mathcal{A}$.*

Next we finish the proof of Theorem 4.1 by bounding the total memory used by $\mathcal{A}_{tr}$.

▶ **Lemma 4.3.** *The average memory used by $\mathcal{A}_{tr}$ is $\mathcal{O}(n\mathtt{I} + n)$.*

Proof. At time t, the expected number of bits appended to the memory (where the expectation is over the choice of $x_{\leq t}$ and $s_{\leq t}$) is bounded by $H(C_t|S_t M_{t-1})$. From (4.2), this is at most $2I(M_t; X_t|M_{t-1}) + \mathcal{O}(1)$. Hence, the number of bits stored in the memory at a time $t \in [n]$ is at most

$$\sum_{i=1}^{t}(2I(M_i; X_i|M_{i-1}) + \mathcal{O}(1)) \leq \sum_{i=1}^{n}(2I(M_i; X_i|M_{i-1}) + \mathcal{O}(1)) = 2n\mathtt{I} + \mathcal{O}(n).$$

Since this is true for every time-step t, the average memory is also upper bounded by $2n\mathtt{I} + \mathcal{O}(n)$. ◀

4.3 Direct Sum for Product Distributions

Recall that we want to prove the following theorem.

▶ **Theorem 4.4** (Direct Sum – Restated). *If p is product input distribution, then*

$$M(f^k) = \Omega\left(k\left(\frac{M(f)}{n} - 1\right)\right).$$

To prove the above we first show that if there is a deterministic algorithm for computing k copies of f with average memory M and error probability $1/3$, then there is a randomized algorithm which computes a single copy of f with error at most $1/3$ and has transitional information content at most M/k. Then, we apply Theorem 4.1 to compress this algorithm and get a contradiction if M is smaller than the right hand side in Theorem 4.4.

4.3.1 Computing f with Small Information

Let $\mathcal{A}$ be a *deterministic* streaming algorithm that uses average memory M and computes f^k on inputs sampled from p^k with error at most $1/3$. Let $m_1, \ldots, m_n$ denote the memory states of the algorithm $\mathcal{A}$. Consider the following *randomized* algorithm $\mathcal{A}_{ran}$ that computes f with error at most $1/3$ on inputs sampled from p. The algorithm chooses a random $j \in [k]$, embeds the input stream at position j and at time t, samples and stores the memory state m_t from the distribution $\mathsf{p}(m_t|x_t^{(j)} = x_t, m_{t-1})$.

Randomized Streaming Algorithm $\mathcal{A}_{ran}$
Input : Stream x sampled from $p(x)$ **Randomness**: j uniformly drawn from $[k]$, streams $x^{(-j)}$ **Output** : $f(x)$ with error at most $1/3$
1. Set Stream $x^{(j)}$ to be x; 2. At time t, use randomness $x_t^{(-j)}$ to sample m_t from $\mathsf{p}(m_t|x_t^{(j)} = x_t, m_{t-1})$; 3. Output the answer of the algorithm on stream j;

Note that for any fixed value of j, the algorithm $\mathcal{A}_{ran}$ uses independent randomness $x_t^{(-j)}$ in each step as the input distribution p is product. We show that on average over the choice of j, the transitional and cumulative information content of the above algorithm is at most M/k.

▶ **Lemma 4.5.** $\mathbb{E}_j[\mathsf{IC}^{tr}(\mathcal{A}_{ran}|J = j)] \leq M/k$ *and* $\mathbb{E}_j[\mathsf{IC}^{cum}(\mathcal{A}_{ran}|J = j)] \leq M/k$.

Proof of Lemma 4.5. Conditioned on any event $J = j$, the transitional information content of $\mathcal{A}_{ran}$ is given by

$$\begin{aligned}
\mathsf{IC}^{tr}(\mathcal{A}_{ran}|J=j) &= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t \mid M_{t-1}, J=j) \\
&= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t^{(j)} \mid M_{t-1}, J=j) \quad \text{(with probability 1, } X^{(j)} = X) \\
&= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t^{(j)} \mid M_{t-1}) \qquad (M_t \text{ ind. of event } J=j).
\end{aligned}$$

Since the input stream comes from a product distribution, $X_t^{(1)}, \ldots, X_t^{(k)}$ are all independent conditioned on M_{t-1}. By Lemma 3.1, the term $I(M_t; X_t^{(j)} \mid M_{t-1})$ in the above sum is bounded by $I(M_t; X_t^{(j)} \mid X_t^{(<j)} M_{t-1})$. Taking an expectation over j, we get

$$\begin{aligned}
\mathbb{E}_j[\mathsf{IC}^{tr}(\mathcal{A}_{ran}|J=j)] &\le \mathbb{E}_j\left(\frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t^{(j)} \mid X_t^{(<j)} M_{t-1})\right) \\
&= \frac{1}{k}\left(\frac{1}{n}\sum_{t=1}^{n}\sum_{j=1}^{k} I(M_t; X_t^{(j)} \mid X_t^{(<j)} M_{t-1})\right)
\end{aligned}$$

From the chain rule the right hand side above equals

$$\frac{1}{k}\left(\frac{1}{n}\sum_{t=1}^{n} I(M_t; X_t^{(1)} \ldots X_t^{(k)} | M_{t-1})\right) = \frac{1}{k}\mathsf{IC}^{tr}(\mathcal{A}) \le \frac{M}{k},$$

where the last inequality follows since the transitional information content is bounded by the average memory (Proposition 3.8).

Analogously, the cumulative information content of $\mathcal{A}_{ran}$ is given by

$$\begin{aligned}
\mathsf{IC}^{cum}(\mathcal{A}_{ran}|J=j) &= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_{\le t} \mid J=j) \\
&= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_{\le t}^{(j)} \mid J=j) \qquad \text{(with probability 1, } X^{(j)} = X) \\
&= \frac{1}{n}\sum_{t=1}^{n} I(M_t; X_{\le t}^{(j)}) \qquad (M_t \text{ind. of event } J=j).
\end{aligned}$$

Since $X^{(1)}, \ldots, X^{(k)}$ are all independent, by Lemma 3.1, the term $I(M_t; X_{\le t}^{(j)})$ is at most $I(M_t; X_{\le t}^{(j)} \mid X_{\le t}^{(<j)})$. Taking an expectation over j and using the chain rule, we get

$$\begin{aligned}
\mathbb{E}_j[\mathsf{IC}^{cum}(\mathcal{A}_{ran}|J=j)] &\le \frac{1}{k}\left(\frac{1}{n}\sum_{t=1}^{n}\sum_{j=1}^{k} I(M_t; X_{\le t}^{(j)} \mid X_{\le t}^{(<j)})\right) \\
&= \frac{1}{k}\left(\frac{1}{n}\sum_{t=1}^{n} I(M_t; X_{\le t}^{(1)} \ldots X_{\le t}^{(k)})\right) = \frac{1}{k}\mathsf{IC}^{cum}(\mathcal{A}) \le \frac{M}{k}. \qquad \blacktriangleleft
\end{aligned}$$

4.3.2 Direct-sum Theorem

With the above, we can now apply Theorem 4.1 to get Theorem 4.4.

Proof of Theorem 4.4. Let $\mathcal{A}$ be a streaming algorithm that computes f^k with error at most 1/3 and average memory M. By Lemma 4.5, there is an algorithm $\mathcal{A}_{ran}$ that uses randomness j and r, computes f with error at most 1/3 and satisfies $\mathbb{E}_j[\mathsf{IC}^{tr}(\mathcal{A}_{ran})|j] \leq M/k$. Applying Theorem 4.1 to $\mathcal{A}_{ran}$ gives us a randomized algorithm that uses random coins j and r' and computes f using average memory $\mathbb{E}_{j,r'}[\frac{1}{n}\sum_{t=1}^{n}|M_t|] = \mathcal{O}(nM/k+n)$.

Since the random coins j and r' are independent of the input, we can fix them to get a deterministic streaming algorithm with average memory $\mathcal{O}(nM/k+n)$. Since this must be at least $M(f)$, we have

$$\mathcal{O}\left(\frac{nM}{k}+n\right) \geq M(f).$$

Rearranging the above gives us that M is lower bounded by $\Omega\left(k\left(\frac{M(f)}{n}-1\right)\right)$. ◀

5 Towards Optimal Direct Sums

The algorithm $\mathcal{A}_{ran}$ that we gave in the last section also had cumulative information content at most M/k as shown in Lemma 4.5. Analogous to Theorem 4.4, the following result follows. We omit the proof since it is very similar to that of Theorem 4.4.

▶ **Theorem 5.1** (Restated). *If every algorithm with cumulative information* $\mathtt{I}$ *can be simulated using average memory* $\mathcal{O}(\mathtt{I})$*, then* $M(f^k) = \Omega\left(k \cdot (M(f)-1)\right)$.

In this section, we describe a compression algorithm that could possibly simulate an algorithm with cumulative information content $\mathtt{I}$ with average memory $\mathcal{O}(\mathtt{I}+1)$. However, we are unable to either prove or disprove it.

To give some intuition about the new algorithm, let us recall Algorithm $\mathcal{A}_{tr}$ where the compression algorithm stored Huffman encodings (Proposition 3.4) of C_t satisfying $X_t - C_tS_t|M_{t-1} \to M_t|M_{t-1}$. This necessitated storing the whole history since to determine the sample m_t required knowing encodings of all the previous $c_{<t}$.

The new algorithm that we call $\mathcal{A}_{cum}$, on receiving the input x_t at time t, samples C_t conditioned on the value of x_t and m_{t-1} where C_t satisfies the following properties that follow from Lemma 3.5:

$$X_t - C_tS_t|S_{<t} \to M_t|S_{<t}, \tag{5.1}$$
$$H(C_t|S_{\leq t}) \leq I(M_t; X_tM_{t-1}|S_{<t}) + \log(I(M_t; X_tM_{t-1}|S_{<t})+1) + \mathcal{O}(1). \tag{5.2}$$

Again the value of m_t determined by the sample c_t is distributed according to the distribution $\mathsf{p}(m_t|x_t, m_{t-1})$. Moreover, the algorithm $\mathcal{A}_{cum}$ will store the Huffman encoding of C_t conditioned on $S_{<t}$ which avoids the need to store all the previous memory contents since $S_{\leq t}$ is randomness independent of the input and can be fixed in the beginning.

Randomized Streaming Algorithm $\mathcal{A}_{cum}$

Input : Stream $x \sim p(x)$
Randomness: $s_1, \ldots, s_n$ where s_i is an infinite sequence of uniform samples from $\mathsf{supp}(M_i) \times [0,1]$.

`// At time` t`: the content of the memory are some encodings of` $c_{<t}$`, where` c_i `determines` m_i `given` $s_{\leq i}$`.`

1. Let m_{t-1} be determined by c_{t-1} and s_{t-1}. On input x_t, sample c_t from the Markov chain in (5.1);
2. Store the Huffman encoding of c_t conditioned on $s_{\leq t}$;

▶ **Conjecture 5.2.** *Let $\mathcal{A}$ be a randomized streaming algorithm with $\mathsf{IC}^{cum}(\mathcal{A}) = \mathtt{I}$. Then, $\mathcal{A}_{cum}$ simulates $\mathcal{A}$ using $\mathcal{O}(\mathtt{I}+1)$ average memory.*

The proof that the above compression algorithm gives a correct simulation is straightforward from (5.1). We are able to prove the following bounds on the memory used by the above algorithm.

▶ **Lemma 5.3.** *In expectation over the choice of $s_{\leq t}$ and $x_{\leq t}$, the memory used by algorithm $\mathcal{A}_{cum}$ at time t is at most $\mathcal{O}(I(M_t; X_{\leq t}|S_{<t}) + 1)$.*

Proof. The memory used by algorithm $\mathcal{A}_{cum}$ at time t is bounded by $H(C_t|S_{\leq t})$ which as given by (5.2) is at most $\mathcal{O}(I(M_t; X_t M_{t-1}|S_{<t})+1)$. Moreover, since $M_{t-1}|S_{<t}$ is determined given $C_{t-1}|S_{<t}$,

$$I(M_t; X_t M_{t-1}|S_{<t}) \leq I(M_t; X_t C_{t-1}|S_{<t}),$$

by the data processing inequality (Proposition 3.2).

Next, we will show that $I(M_t; X_t C_{t-1}|S_{<t})$ is upper bounded by $I(M_t; X_{\leq t}|S_{<t})$. To show this we bound $I(M_t; X_t C_{t-1}|S_{<t}) \leq I(M_t; X_{\leq t} C_{t-1}|S_{<t})$ which by chain rule is,

$$\begin{aligned} &= I(M_t; X_{<t} C_{t-1}|S_{<t}) + I(M_t; X_t|S_{<t} X_{<t} C_{t-1}) \\ &= I(M_t; X_{<t}|S_{<t}) + I(M_t; C_{t-1}|S_{<t} X_{<t}) + I(M_t; X_t|S_{<t} X_{<t} C_{t-1}). \end{aligned}$$

Note that in the algorithm $\mathcal{A}_{cum}$, $X_{<t}$ and $S_{<t}$ completely determine C_{t-1}. Hence, the second term in the above expression is 0. Moreover, by the same fact $M_t X_t - S_{<t} X_{<t} \to C_{t-1}$ and hence by Proposition 3.3, the last term $I(M_t; X_t|S_{<t} X_{<t} C_{t-1}) = I(M_t; X_t|X_{<t} S_{<t})$.

The above discussion yields that

$$\begin{aligned} I(M_t; X_t C_{t-1}|S_{<t}) &\leq I(M_t; X_{<t}|S_{<t}) + I(M_t; X_t|X_{<t} S_{<t}) \\ &= I(M_t; X_{\leq t}|S_{<t}), \end{aligned}$$

where the second equality follows by another application of the chain rule. ◀

Note that since $S_{<t}$ is independent of $X_{\leq t}$, the above quantity $I(M_t; X_{\leq t}|S_{<t})$ is at least as large as $I(M_t; X_{\leq t})$ (recall Lemma 3.1), but it is possible that similar to Lemma 3.5, they could be the same up to some lower order error terms. Towards proving such a statement, we first propose to investigate whether the following stronger version of Lemma 3.5 holds.

▶ **Conjecture 5.4.** *Let X and M be arbitrary discrete random variables with finite support. Let S be an infinite list of samples from $\mathsf{supp}(M) \times [0,1]$. Then, there exist a random variable C such that*

- $X - CS \to M$.
- $H(C|S) \leq I(M; X) + \log(I(M; X) + 1) + \mathcal{O}(1)$.
- *For any discrete random variable N such that $X - M - N$, it holds that*

$$I(N; M|S) \leq I(N; X) + \log(I(N; X) + 1) + \mathcal{O}(1).$$

We also point out that an inductive use of the above conjecture does not give a non-trivial upper bound on the memory used by the algorithm $\mathcal{A}_{cum}$ because of the error terms in the last statement of the conjecture. But we hope that the techniques used in proving the above conjecture would be helpful in analyzing the memory used by the algorithm $\mathcal{A}_{cum}$. Nonetheless the above conjecture might be interesting in its own right and of potential use somewhere else.

Acknowledgments. We thank Paul Beame for helpful comments.

References

1. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
2. Chrisil Arackaparambil, Joshua Brody, and Amit Chakrabarti. Functional monitoring without monotonicity. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 95–106. Springer Berlin Heidelberg, 2009.
3. Brian Babcock and Chris Olston. Distributed top-k monitoring. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD'03, pages 28–39, New York, NY, USA, 2003. ACM.
4. Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An Information Statistics Approach to Data Stream and Communication Complexity. In *FOCS*, pages 209–218, 2002.
5. Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013.
6. Mark Braverman. Interactive information complexity. In *STOC*, pages 505–524, 2012.
7. Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. A tight bound for set disjointness in the message-passing model. In *FOCS*, pages 668–677, 2013.
8. Mark Braverman and Ankit Garg. Public vs Private Coin in Bounded-Round Information. In *ICALP*, pages 502–513, 2014.
9. Mark Braverman and Anup Rao. Information equals amortized communication. In *FOCS*, pages 748–757, 2011.
10. Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, pages 107–117, 2003.
11. Arkadev Chattopadhyay, Jaikumar Radhakrishnan, and Atri Rudra. Topology matters in communication. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 631–640, 2014.
12. G. Cormode, S. Muthukrishnan, and Wei Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 57–57, April 2006.
13. Graham Cormode. Sketching streams through the net: Distributed approximate query tracking. In *VLDB*, pages 13–24, 2005.
14. Graham Cormode and Minos Garofalakis. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *SIGMOD*, pages 25–36, 2005.
15. Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. *ACM Trans. Algorithms*, 7(2):21:1–21:20, March 2011.
16. Graham Cormode, S. Muthukrishnan, Ke Yi, and Qin Zhang. Optimal sampling from distributed streams. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS'10, pages 77–86, New York, NY, USA, 2010. ACM.
17. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.
18. Pavol Duris and Jose D.P. Rolim. Lower bounds on the multiparty communication complexity. *Journal of Computer and System Sciences*, 56(1):90–95, 1998.

19 Funda Ergun and Hossein Jowhari. On distance to monotonicity and longest increasing subsequence of a data stream. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'08, pages 730–736, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

20 Anna Gál and Parikshit Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *SIAM J. Comput.*, 39(8):3463–3479, August 2010.

21 Anat Ganor, Gillat Kol, and Ran Raz. Exponential Separation of Information and Communication for Boolean Functions. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:113, 2014.

22 André Gronemeier. Asymptotically optimal lower bounds on the nih-multi-party information complexity of the and-function and disjointness. In *STACS 2009*, pages 505–516, 2009.

23 Sudipto Guha and Zhiyi Huang. Revisiting the direct sum theorem and space lower bounds in random order streams. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 513–524. Springer Berlin Heidelberg, 2009.

24 Prahladh Harsha, Rahul Jain, David A. McAllester, and Jaikumar Radhakrishnan. The communication complexity of correlation. *IEEE Transactions on Information Theory*, 56(1):438–449, 2010. doi:10.1109/TIT.2009.2034824.

25 Zengfeng Huang, Božidar Radunović, Milan Vojnović, and Qin Zhang. Communication complexity of approximate maximum matching in distributed graph data. In *STACS*, 2015.

26 Ram Keralapura, Graham Cormode, and Jeyashankher Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD'06, pages 289–300, New York, NY, USA, 2006. ACM.

27 Amit Manjhi, Vladislav Shkapenyuk, Kedar Dhamdhere, and Christopher Olston. Finding (recently) frequent items in distributed data streams. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE'05, pages 767–778, Washington, DC, USA, 2005. IEEE Computer Society.

28 Marco Molinaro, David P. Woodruff, and Grigory Yaroslavtsev. Beating the direct sum theorem in communication complexity with implications for sketching. In *SODA*, pages 1738–1756, 2013.

29 Jeff M. Phillips, Elad Verbin, and Qin Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'12, pages 486–501. SIAM, 2012.

30 Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems*, 32(4), November 2007.

31 Izchak Sharfman, Assaf Schuster, and Daniel Keren. Shape sensitive geometric monitoring. In *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS'08, pages 301–310, New York, NY, USA, 2008. ACM.

32 David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'04, pages 167–175, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

33 David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *STOC*, pages 941–960, 2012.

34 David P. Woodruff and Qin Zhang. An optimal lower bound for distinct elements in the message passing model. In *SODA*, pages 718–733, 2014.
35 A.D. Wyner. The common information of two dependent random variables. *IEEE Transactions on Information Theory*, 21(2):163–179, 1975.

A Separation between common information and mutual information

In this section we will give an explicit example of random variables X and M such that $\mathbb{C}(X;M) = \omega(I(X;M))$. Let G be a bipartite graph on the vertex set $([n],[n])$ such that the edge density of G is $\frac{1}{2} + o(1)$ and there are no cliques with more than $3n \log n$ edges in G. As the following lemma shows a random bipartite graph where each edge is picked with probability $1/2$ satisfies these properties with high probability, so such graphs exist.

▶ **Lemma 1.1.** *With probability* $1 - o(1)$*, a random bipartite graph on* $([n],[n])$ *where each edge is included with probability* $1/2$ *has no clique* $U \times V$ *where* $U, V \subseteq [n]$ *satisfying* $\min\{|U|,|V|\} \geq 2\log n + 2$.

Proof. Set $t := 2\log n + 2$ for notational convenience. If there is a clique $U \times V$ with $\min\{|U|,|V|\} \geq t$ then there also exists a clique of size $t \times t$. Consequently, to prove the lemma it suffices to upper bound the probability that a $t \times t$ clique exists in the graph. This probability is at most

$$\binom{n}{t}\binom{n}{t} 2^{-t^2} \leq n^{2t} 2^{-t^2} = 2^{2t\log n - t^2} = 2^{t(2\log n - t)} \leq 2^{-2t} = o(1). \quad ◀$$

A corollary of the above lemma is that the maximal clique in a random bipartite graph with edge probability $1/2$ has at most $n \cdot 3\log n$ edges with high probability. Also it is easy to see that with probability $1 - o(1)$, every vertex in a random bipartite graph with edge probability $1/2$ has degree between $\frac{n}{2} - o(n)$ and $\frac{n}{2} + o(n)$.

Now we can describe the random variables X and M which will be the end points of a uniformly random edge E in the graph G. It is easily seen that the mutual information $I(X;M) \leq 1 - o(1)$ since $H(X) = \log n$ while for any $M = m$, $H(X|M = m) \geq \log n - 1 - o(1)$. On the other hand, if $X - W - M$, then for any value w attained by W, $\mathsf{supp}(X|W = w)$ and $\mathsf{supp}(M|W = w)$ has to form a clique in the graph G. Since the maximal clique in G has at most $3n \log n$ edges, for any $W = w$, it holds that

$$|\mathsf{supp}(X|W = w)| \cdot |\mathsf{supp}(M|W = w)| \leq 3n \log n.$$

It follows that for any such W we can write

$$H(XM|W) \leq \log(|\mathsf{supp}(X|W = w)| \cdot |\mathsf{supp}(M|W = w)|) = \log n + \mathcal{O}(\log\log n).$$

Hence we have that the mutual information between XM and W is,

$$\begin{aligned} I(XM;W) &= H(XM) - H(XM|W) \geq (2\log n - 1 - o(1)) - (\log n + \mathcal{O}(\log\log n)) \\ &= \log n - \mathcal{O}(\log\log n), \end{aligned}$$

for any W satisfying $X - W - M$. It follows that $\mathbb{C}(X;M) = \Omega(\log n)$ while $I(X;M) \leq 1 - o(1)$.