Electronic Colloquium on Computational Complexity, Report No. 79 (2007)

One-way multi-party communication lower bound for pointer jumping with applications

Emanuele Viola* Avi Wigderson†

August 7, 2007

Abstract

In this paper we study the one-way multi-party communication model, in which every party speaks exactly once in its turn. For every fixed k, we prove a tight lower bound of $\Omega\left(n^{1/(k-1)}\right)$ on the probabilistic communication complexity of pointer jumping in a k-layered tree, where the pointers of the i-th layer reside on the forehead of the i-th party to speak. The lower bound remains nontrivial even for $k = (\log n)^{1/2-\Omega(1)}$ parties. Previous to our work a lower bound was known only for $k = 3$ [BHK], and in very restricted models for $k > 3$ [DJS, Cha]. Our results have the following consequences to other models and problems, extending previous work in several directions.

The one-way model is strong enough to capture *general* (non one-way) multi-party protocols of bounded rounds. Thus we generalize to this multi-party model results on two directions studied in the classical 2-party model (e.g. [PS, NW]). The first is a round hierarchy: We give an exponential separation between the power of r and $2r$ rounds in general probabilistic k-party protocols, for any fixed k and r. The second is the relative power of determinism and nondeterminism: We prove an exponential separation between nondeterministic and deterministic communication complexity for general k-party protocols with r rounds, for any fixed k, r.

The pointer jumping function is weak enough to be a special case of the well-studied disjointness function. Thus we obtain a lower bound of $\Omega\left(n^{1/(k-1)}\right)$ on the probabilistic complexity of k-set disjointness in the one-way model, which was known only for $k = 3$ parties. Our result also extends a similar lower bound for the weaker simultaneous model, in which parties simultaneously send one message to a referee [BPSW].

Finally, we infer an exponential separation between the power of different orders in which parties send messages in the one-way model, for every fixed k. Previous to our work such a separation was only known for $k = 3$ [NW].

Our lower bound technique, which handles functions of high discrepancy, may be of independent interest. It provides a "party-elimination" induction, based on a restricted form of a direct-product result, specific to the pointer jumping function.

*The author is supported by NSF grant CCR-0324906.
†The author is supported by NSF grant CCR-0324906.

ISSN 1433-8092

1 Introduction

In Yao's standard communication complexity [Yao], two parties are each holding an input, and they attempt to compute (or approximate) a given function of these two inputs by exchanging at most c bits of communication (cf. the comprehensive book [KN]). This model has been one of the most extensively studied in computational complexity, and captures essential features of many diverse computational settings, from Turing machines, VLSI and distributed computation, to linear programming and auctions. Many techniques for proving various strong lower bounds are known.

This model was generalized by Chandra, Furst, and Lipton [CFL], to the *multiparty model* (often called "number-on-forehead"). In k-party communication complexity each party is assigned an input again, and the parties have to compute (or approximate) a function $f : X_1 \times X_2 \times \cdots \times X_k = (\{0,1\}^n)^k \to \{0,1\}$ of all k inputs by exchanging c bits of communication. However, the input $x_i \in X_i$ to the i-th party (figuratively) resides on that party's forehead, and so (formally) each party knows *all but* his own input. The overlapping information of the parties allows this model to capture more complex settings, like multi-tape Turing machines, branching programs, constant-depth circuits with modular gates and more. In the multiparty model, lower bounds are known for as many as $k = k(n) = \epsilon \cdot \log n$ parties. We point out that even for the most restricted form of this model, namely *simultaneous* multi-party protocols (in which all parties send one message to a referee) no explicit function is known which cannot be computed using $k = \log_2 n$ parties and $c = \log_2 n$ communication. Crossing this barrier will have impressive complexity implications, but has so far resisted all attacks.

In this paper we obtain several new results on natural variants and questions in the multi-party model, which we describe in the next subsections with their background.

1.1 Rounds in communication complexity.

Papadimitriou and Sipser [PS] initiated the study of how 2-party communication complexity is affected by limiting the two parties to only r rounds of messages. This natural question was taken on by several researchers, who obtained exponential separation between r and $r+1$ rounds in 2-party protocols (see [NW] and the references within). No separation was known for $k > 2$ parties. In this work we obtain the first such result, stated next.

Theorem 1.1 (Round hierarchy)**.** *For every fixed constants r and k there is an explicit function $f : (\{0,1\}^n)^k \to \{0,1\}$ that requires $n^{\Omega(1)}$ communication for k-party r-rounds protocols, but that can be computed with $O(\log n)$ communication by k-party r'-rounds protocols, if $r'(k-1) \geq r \cdot k$.*

For k-party r-rounds protocols we also obtain an exponential separation between *nondeterministic* and deterministic protocols. Again, this seems to be the first such separation for $k > 2$ parties.

Theorem 1.2 (Nondeterminism vs. determinism)**.** *For every fixed constants r and k there is an explicit function $f : (\{0,1\}^n)^k \to \{0,1\}$ that requires $n^{\Omega(1)}$ communication for k-party r-rounds protocols, but that can be computed with $O(\log n)$ communication by nondeterministic k-party 2-rounds protocols.*

The above results are obtained as a corollary from a new communication complexity lower bound in the one-way model, which we describe next.

1.2 The one-way model and pointer jumping.

In the one-way model the k-parties speak in turn, and only once. This is an interesting model that arises in several contexts, such as oblivious branching programs and streaming algorithms [AMS, BYJKS]. In this work we establish a new one-way lower bound for the *pointer jumping* function. To define this function, we think of a regular tree[1] of depth k. The input to the pointer jumping function specifies a $\{0,1\}$-value for the leaves, and for every internal node a pointer to one of its children. The output of the pointer jumping function is the bit obtained by following the pointers from the root to a leaf. We refer the reader to Figure 1 on Page 5 for a picture. Party i knows all pointers except those from the i-th layer to the $(i+1)$-th.

The pointer jumping function naturally captures "inherently sequential" computation and inference. It has been studied by many researchers in many models and contexts, including constant-depth circuits, proof complexity, PRAMs and communication complexity (e.g. [PS, NW, NBY, BIP, DJS, BHK, Cha])

In particular, attempts to proving lower bounds in the one-way multi-party model have existed for over a decade. For $k = 3$ this was achieved by Wigderson (cf. appendix in [BHK]), who proved a $\Omega(\sqrt{n})$ lower bound via extremal set theory arguments. For $k > 3$ strong lower bounds were obtained only for various (unnatural) restrictions of the one-way model (the "conservative" model and the "myopic" model) [DJS, Cha]. The main result of this paper makes progress in the general one-way model, obtaining a tight lower bound for any constant number k of parties. Our lower bound remains nontrivial for any number of parties $k = (\log n)^{1/2-\Omega(1)}$.

Theorem 1.3. *For every k, the pointer jumping function $TPJ_k : (\{0,1\}^n)^k \to \{0,1\}$ requires communication at least $\left(n^{1/(k-1)}\right)/k^{O(k)}$ for one-way k-party number-on-forehead protocols.*

Another consequence of our Theorem 1.3 is about the relative power of one-way k-party protocols with respect to various *orders* in which the parties speak. Exponential separations between different orders were again given by Nisan and Wigderson [NW, BHK] for the case $k = 3$, but were unknown for larger k. As a corollary of Theorem 1.3 we obtain an exponential separation for every constant k:

Corollary 1.4 (Order separation)**.** *For any fixed constant k and any order π for the k parties to speak, there is an explicit function $f : (\{0,1\}^n)^k \to \{0,1\}$ that requires $n^{\Omega(1)}$ communication if the parties speak in the order π, but can be computed with $O(\log n)$ communication if the parties speak in any other order.*

1.3 Discrepancy and disjointness.

Another reason for our interest in the one-way model is that, for some basic functions, it is the strongest model in which multiparty communication complexity lower bounds are known. Specifically, lower bounds in the general model are typically obtained by upper bounding the *discrepancy* of the target function, a quantity which measures the bias of the function over large cylinder intersections. Although we know of functions with exponentially small discrepancy, such as the generalized inner product [BNS, CT, Raz, VW], some important functions in fact have high discrepancy, and consequently the above technique cannot be applied to prove communication complexity lower bounds. The pointer jumping function, described above, is an example of such a function.

[1] It makes sense to consider this function over arbitrary graphs, but working with trees is more convenient for our purposes – jumping ahead, it will more directly implies a lower bound for disjointness.

Another fundamental example of a function with high discrepancy is the *disjointness* function. Here, each of the k parties is assigned a subset of $\{1, 2, \ldots, n\}$, and the parties wish to determine whether or not their sets share a common element. The study of the disjointness function is central to communication complexity, and is motivated both by the naturalness of the function and by its interesting consequences in proof complexity and circuit complexity. For example, in propositional proof complexity, it is shown in [BPS] that sufficiently strong ($c = \omega(\log^4 n)$) lower bounds for the k-party disjointness would solve the major open problem of obtaining proof size lower bounds for a family of proof systems which capture certain semi-definite optimization techniques, and are known as tree-like, degree $k-1$ threshold systems. In circuit complexity, it can be shown using the techniques in [RW, Vio] that sufficiently strong ($\log^d n$ for a certain constant d) lower bounds for the $\Omega(\log n)$-party disjointness function would separate polynomial-size constant-depth circuits from polynomial-size depth-2 circuits with an arbitrary symmetric gate at the root.

The difficulty of obtaining such lower bounds for the general multi-party model has caused researchers to try more restricted models first. For disjointness, a $\Omega\left(n^{1/(k-1)}\right)$ lower bound was known in the simultaneous model [BPSW], where there is no interaction between parties. In the stronger one-way model a $n^{\Omega(1)}$ lower bound was only known for $k = 3$ (cf. appendix in [BHK] and [BPSW]).[2] In this work we extend these previous results by obtaining a $\Omega\left(n^{1/(k-1)}\right)$ lower bound for one-way protocols.

Corollary 1.5. *The disjointness function $DISJ_{k,n} : (\{0,1\}^n)^k \to \{0,1\}$ requires communication at least $\left(n^{1/(k-1)}\right)/k^{O(k)}$ for one-way k-party number-on-forehead protocols.*

We note that this result is obtained as a corollary of our lower bound for pointer jumping. It is easy to see that when the underlying graph is a tree, pointer jumping is a special case of disjointness.

1.4 Techniques

We now summarize how we obtain our lower bound for pointer jumping (Theorem 1.3). The bound is obtained by induction on the number of parties k. Carrying through this induction is somewhat involved, and in particular we will need to consider non-Boolean variants of pointer jumping, in which one has to chase pointers on several trees. Specifically, we will work with the function $TPJ_k^{m,d} : (\{0,1\}^n)^k \to \{0,1\}^m$, which we think of as a forest of m d-regular trees of depth k. Again, the input to $TPJ_k^{m,d}$ specifies a $\{0,1\}$-value for the leaves, and for every internal node a pointer to one of its d children. The output of the pointer jumping function is the m-bit string obtained by following the paths from the m roots to the leaves. It will be important for us to show that computing $TPJ_k^{m,d}$ is even difficult if we allow for success probability that is subexponentially small in m (in the Boolean $m = 1$ case this probability bound is 0.99, say).

Specifically, we prove by induction on k that the following statement holds for every choice of m and d (cf. Theorem 3.2), which may be viewed as a direct product theorem in this model, specific to pointer jumping:

> (I) No k-party one-way protocol using communication $o(m \cdot d)$ can compute $TPJ_k^{m,d}$ with probability $\exp(-o(m))$ over the choice of a random input.

[2] We note that in [BPSW] they also prove a $n^{\Omega(1)}$ bound for 3-party disjointness in a stronger model where the first party speaks once, and then the two other parties interact arbitrarily.

Proving the inductive step in (I) is the main technical contribution of this work. To this aim we develop our main *party reduction* Theorem 3.3. This theorem shows how to transform a given protocol for $TPJ_k^{m,d}$ into another protocol for pointer jumping with $k-1$ parties on $m' := d \cdot m$ trees, $TPJ_{k-1}^{m',d}$. For example, in the special case in which we start with a Boolean ($m = 1$) function with k parties, we construct a protocol for a non-Boolean ($m' = d$) function with $k-1$ parties. As we point out later, our party reduction theorem in fact gives a general transformation for protocols in which the first party speaks only once, and thus it applies to other models beyond one-way, and other functions besides pointer jumping.

This transformation is obtained as follows. We fix a message for the first party that maximizes the success probability of the protocol. Thinking of the input as $(x_1, y) \in X_1 \times Y$, where $Y = X_2 \times \cdots X_k$, we note that such fixing decreases the probability of success over the choice of y (however, jumping ahead, this loss will be $\exp(-o(d \cdot m)) = \exp(-o(m'))$ which is fine for (I)), but does not decrease the success probability over the choice of x_1, because the message does not depend on x_1. This bigger success probability over x_1 allows us to run the protocol $r \approx d$ times for different choices of pointers $x_1 \in X_1$ and still tolerate the loss in the error (recall that $d = m'/m$). At this point, we argue that the r outputs of the protocol let us compute $(1 - o(1)) \cdot m'$ bits of the m' bits of $TPJ_{k-1}^{m',d}$ that we need. To argue this, we proceed by showing that only with probability exponentially small the pointers (given by all r runs of the protocol) hit less than $(1 - o(1)) \cdot m'$ bits. By a suitable choice of parameters, this probability can be made smaller than the probability that all r runs of the protocol are correct. Thus, with sufficiently high probability, all r runs of the protocol are correct *and* their accumulated pointers hit at least $(1 - o(1)) \cdot m'$ bits. When this happens, we can compute each of these $(1 - o(1)) \cdot m'$ bits by outputting the corresponding output bit of the protocol.

It remains to deal with the $o(1)m'$ bits we know nothing about. To compute these, we simply guess at random. The probability of guessing them correct will be $\exp(-o(m'))$, which is still fine for (I). This concludes our proof overview (cf. Remark 4.3 for the history of this approach).

To clarify the parameters in the above discussion we remark that, intuitively, $(1 - o(1)) \cdot m'$ bits is as much information as we can hope to obtain from the protocol: There may well be a fraction of $o(1)m'$ indices such that the protocol never gives a meaningful answer if a pointer falls there; such a protocol would satisfy the inductive hypothesis because its success probability would be $\exp(-o(m))$, but its outputs give no information on the value of $o(1)m'$ bits.

Finally, we point out that the idea of employing a party-reduction theorem is already present in the work [DJS], where is used to derive lower bounds in communication models that are more restricted than the one-way model considered here. Also, a related round-reduction idea can be found in the 2-party lower bound in [NW].

Organization. This paper is organized as follows. After some preliminaries in Section 2, in Section 3 we formally state our lower bounds for pointer jumping, including our main party reduction theorem, and then we derive our lower bound assuming the party-reduction theorem. In Section 4 we prove the party-reduction theorem. The consequences of our lower bound for general protocols with bounded number of rounds (Theorems 1.1 and 1.2), relative power of different orders in the one-way model (Corollary 1.4), and disjointness (Corollary 1.5), are given in Section 5. Finally, in Section 6 we discuss some applications of our results to streaming models of computation.

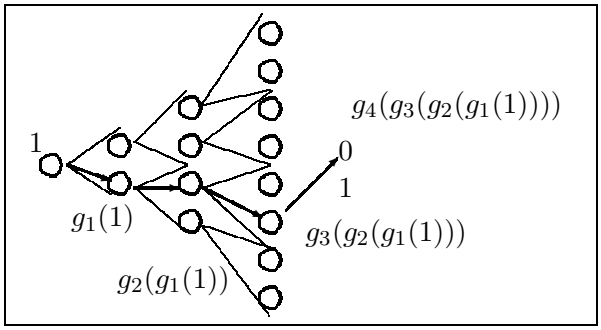

Figure 1: TPJ function ($k = 4, m = 1, d = 2$).

2 Background on communication complexity

We use standard definitions of communication protocols, which we now briefly recall. In a k-party protocols, k parties with to collaboratively compute a function $f : X_1 \times X_2 \times \cdots \times X_k \to D$ on input $x = (x_1, x_2, \ldots, x_k)$. The function f is known to each party, and the i-th party sees all pieces of x *except* x_i. The parties communicate by broadcasting messages, and the final party is supposed to communicate the output of the function. The communication complexity of a protocol is the total length of the messages exchanged; note that we count the final message (which if the protocol is correct is the value of the function) towards communication.

In a r-round protocol, the parties speak in any order $w \in [k]^r$, which is chosen adaptively during the protocol: At any round, the party to speak is a function of the messages exchanged up to that round. We stress that r may be much larger than k, and that parties may speak more than once. We now define more restricted models where the order is fixed a priori, i.e. the parties always speak in the same order regardless of the messages exchanged. A *one-way protocol with order* w is a $(r = k)$-round protocol in which the parties always speak in the order w, and w is a permutation of $[k]$. A (standard) *one-way* protocol is a one-way protocol with order w where w is the identity permutation $w = 1, 2, 3, \ldots, k$.

For concreteness, we note that a 1-party one-way protocol $P : X_1 \to \{0,1\}^m$ is simply a fixed string $P(x_1) = P \in \{0,1\}^m$, independent of the input. A 2-party $c-$bit one-way protocol $P : X_1 \times X_2 \to \{0,1\}^m$ is given by two functions f_1 and f_2, where $P(x_1, x_2) = f_2(f_1(x_2), x_1)$, and the output length of each function is bounded by c. The function f_1 computes the message of the first party, which depends on x_2 only, while f_2 computes the (output) message of the second party, which depends on x_1 and the message $f_1(x_2)$ sent by the first party. Similarly, a 3-party c-bit one-way protocol $P : X_1 \times X_2 \times X_3 \to \{0,1\}^m$ is given by three functions f_1, f_2, and f_3, where $P(x_1, x_2, x_3) = f_3(f_2(f_1(x_2, x_3), x_1, x_3), x_1, x_2)$, and the output length of each function is bounded by c.

A *probabilistic* or *randomized* protocol is a probability distribution over deterministic protocols.

3 Lower bound for pointer jumping

We now define the pointer jumping function and then state our main result. The pointer jumping function can be defined in different ways, and in this work we adopt the following variant. We

think of a forest of m d-regular trees of depth k. The input to the pointer jumping function specifies a $\{0,1\}$-value for the leaves, and for every internal node a pointer to one of its d children. (Specifically, for every level the j-th internal node at that level has its d children labelled from the interval $[d \cdot (j-1)+1, d \cdot j]$.) The output of the pointer jumping function is the m-bit string obtained by following the paths from the m roots to the leaves. We refer the reader to Figure 1 on Page 5 for a picture. Compared to other definitions (e.g., [BHK]), ours has the advantage of having a shorter input length (thus giving a stronger lower bound), and more directly implying a lower bound for disjointness (cf. Corollary 5.3).

Definition 3.1 (Pointer jumping function)**.** *For integers k, m, d, the pointer jumping function is*

$$TPJ_k^{m,d} : [d]^m \times [d]^{m \cdot d} \times \cdots [d]^{m \cdot d^2} \times \{0,1\}^{m \cdot d^{k-1}} \to \{0,1\}^m,$$

where for every i, and for $j \in [m]$ we have

$$TPJ_k^{m,d}(g_1, g_2, \ldots, g_k)_j := (g_k \circ g_{k-1} \circ \cdots \circ g_1)(j) = g_k(g_{k-1}(\cdots(g_1(j))\cdots),$$

where $g_i \in BF_d^{m \cdot d^{i-1}} := \left\{g : [m \cdot d^{i-1}] \to [m \cdot d^i] \text{ such that } g(j) \in [d \cdot (j-1)+1, d \cdot j]\right\}$ for $i < k$, and $g_k : [m \cdot d^{k-1}] \to \{0,1\}$.

Note that the function $TPJ_k^{m,d}$ has input length $n = m \cdot (d^{k-1} + d^{k-2} \cdot \log d + d^{k-3} \cdot \log d + \cdots + d^0 \cdot \log d) = O(m \cdot d^{k-1})$. This function is Boolean for $m = 1$, while for $m > 1$ its output is a m-bit string corresponding to the outputs of m Boolean pointer jumping functions on independent inputs.

We now state our lower bound for pointer jumping. Here and throughout the paper we let $\exp(x) := 2^x$ and $\log = \log_2$. Also, all probabilities are taken over the uniform distribution.

Theorem 3.2. *For every $k, m, d \geq 1$, and for $\mu_k := (k+1)^{-100k}$, there is no one-way k-party protocol P such that*

$$\Pr_{x \in X}\left[P(x) = TPJ_k^{m,d}(x)\right] \geq \exp(-\mu_k \cdot m)$$

and P uses communication at most $\mu_k \cdot m \cdot d$.

In particular, every one-way k-party protocol for the Boolean pointer jumping function $TPJ_k^{1,d} : \{0,1\}^n \to \{0,1\}$ must use communication at least $d/k^{O(k)} = \left(n^{1/(k-1)}\right)/k^{O(k)}$.

Note that, in particular, the above theorem gives a lower bound on the communication which is $\exp\left(\log^{\Omega(1)} n\right)$ for every $k \leq \log^{1/2-\Omega(1)} n$. Also note that the $TPJ_k^{1,d}$ function can be computed with $d+1$ bits of communication – simply by letting the first party communicate the d possible outputs of the function for each value of the input on her forehead, and then letting the second party announce the output of the function. Thus, for constant k our lower bound is tight up to constant factors; but in general we do not know whether our $d/k^{O(k)}$ bound is tight or not. Finally, we note that the bound in the above theorem is *distributional*, i.e. it limits the fraction of inputs on which any low-communication protocol can compute $TPJ_k^{m,d}$ correctly. By an averaging argument (see, e.g., [KN, Theorem 3.20]), this implies a lower bound for randomized protocols as well.

The basic intuition for the proof of Theorem 3.2 is that if we can solve m problems correctly (i.e., compute the m output bits of $TPJ_k^{m,d}$) with some success probability and communication, then by repeating the protocol roughly d times we should be able to solve $m \cdot d = m'$ problems (i.e.,

compute the m' output bits of $TPJ_{k-1}^{m',d}$) with loss of a factor d in communication and exponential in d in success probability.[3]

Inducting on the above will give a contradiction. Specifically, the proof of Theorem 3.2 proceeds by induction on k. For every k, we prove the theorem for every setting of the other parameters. For the inductive step, we show how to turn a given protocol for the function with k parties into another protocol for the function with $k-1$ parties on a bigger forest (for example, in the special case in which we start with a Boolean function with k parties, we construct a protocol for a non-Boolean function with $k-1$ parties). This transformation is provided by the following main Theorem 3.3. Even though in this work we mainly apply this theorem (3.3) to one-way protocols, it applies to models beyond one-way multiparty communication: It gives a generic reduction which transforms any protocol with one special party who speaks first and only once, into a related protocol without this party. Thus a "protocol" may be specified in different ways, and the relation between the given and the new protocol (as well as what they compute) is specified formally in the theorem.

Theorem 3.3 (Main – party reduction)**.** *Let m, d be integers, and $m' := d \cdot m$. Let $f' : X \to \{0,1\}^{m'}$ be any function, and consider the function $f : BF_d^m \times X \to \{0,1\}^m$ defined as $f(g,x)_j := f'(x)_{g(j)}$ where $g \in BF_d^m$.*

Suppose that there is a protocol $P(g,x)$ using communication $\mu \cdot m \cdot d$ such that:

$$\Pr_{g \in BF_d^m, x \in X}[P(g,x) = f(g,x)] \geq \exp(-\mu \cdot m),$$

where g is on the forehead of a party who speaks first and only once, and $\mu < 1/2$.

Then there is a protocol $P'(x)$ using communication $\mu' \cdot m' \cdot d$ such that

$$\Pr_{x \in X}\left[P'(x) = f'(x)\right] \geq \exp(-\mu' \cdot m'),$$

where

$$\mu' = 100\mu \cdot \log(1/\mu)$$

and P' can be written as $P'(x) = h\left(P(\tilde{g}_1, x), P(\tilde{g}_1, x), \ldots, P(\tilde{g}_r, x)\right)$ for some fixed function $h : \{0,1\}^{m \cdot r} \to \{0,1\}^{m'}$ and fixed inputs $\tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_r \in BF_d^m$ (and some r). In particular, if $X = X_2 \times \cdots \times X_k$ and P is a one-way k-party protocol, then P' is a one-way $(k-1)$-party protocol.

Proof of Theorem 3.2 from Theorem 3.3. To get a sense of the parameters, note that the statement that we set to prove is monotone in μ_k. Specifically, for any fixed k, m, and d, if there is no protocol P such that $\Pr_{x \in X}\left[P(x) = TPJ_k^{m,d}(x)\right] \geq \exp(-\mu_k \cdot m)$ and P uses communication $\mu_k \cdot m \cdot d$, where $\mu_k = \alpha$, then the same is true for any choice of smaller $\mu_k := \beta < \alpha$.

We proceed by induction on k.

Base case $k = 1$. In this case the communication does not play a role and P is a fixed string $P \in \{0,1\}^m$, whereas $TPJ_1^{m,d}(x) = x$. The probability that a random input m-bit string x equals a fixed string P is $\exp(-m) < \exp(-\mu_1 \cdot m)$, and thus the base case is proved.

[3] Our initial version of this argument was, intuitively, that being able to compute *most* of the m output bits of $TPJ_k^{m,d}$ implies the ability to compute *most* of the m' output bits of $TPJ_{k-1}^{m',d}$. The proof of this was quite complex, and the simplification here derives from an idea in [BRW] which was pointed out to us by Oded Regev. We will stress in the proof where their idea is used.

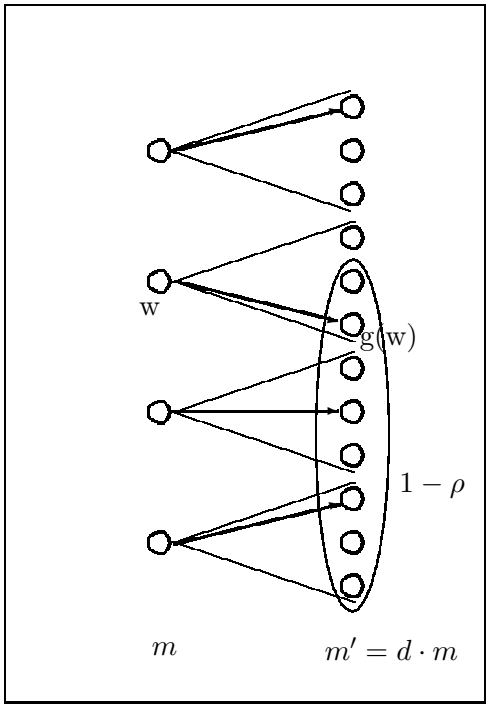

Figure 2: Variables of the proof of Theorem 3.3.

Inductive step. We prove the inductive step $k \geq 2$ by contradiction. Suppose that there is a protocol for $TPJ_k^{m,d}$ with parameter $\mu_k = (k+1)^{-100k} < 1/2$. We apply Theorem 3.3 to obtain a protocol for $TPJ_{k-1}^{d \cdot m,d}$ with parameter μ_{k-1} which equals

$$100\mu_k \cdot \log(1/\mu_k) = 100(k+1)^{-100k} \cdot 100k \cdot \log(k+1)$$
$$\leq (k+1)^{-100k} \cdot k^{100} \leq k^{-100k} \cdot k^{100} = k^{-100(k-1)}.$$

This contradicts the inductive hypothesis for $(k-1)$.

The "in particular" part of the theorem follows from the easily verifiable fact that the input length of $TPJ_k^{1,d}$ is $n = O(d^{k-1})$ (cf. the discussion after Definition 3.1). □

In the next section we prove Theorem 3.3. The reader may want to refer to Figure 2 while reading the proof.

4 Proof of the party-reduction Theorem 3.3

We assume without loss of generality that $d \geq 1/\mu$. This is because if $d < 1/\mu$ then the protocol P is using less than m bits of communication, and since we count the output length of P towards communication, the protocol cannot even output an m-bit string, in which case the hypothesis of the theorem is always false and the theorem vacuously true.

From the hypothesis and a Markov argument[4] it follows that

$$\Pr_{x\in X}\left[\Pr_{g\in BF_d^m}[P(g,x)=f(g,x)]\geq \exp(-2\mu\cdot m)\right]\geq \exp(-\mu\cdot m)/(1+\exp(-\mu\cdot m))\geq \exp(-\mu\cdot m)/2.$$

The message sent by the first party in P does not depend on g, since g lies on the forehead of the first party. Using this and the fact that the communication of P is $\mu\cdot m\cdot d=\mu\cdot m'$, we can fix a particular message a for the first party such that

$$\Pr_{x\in X}\left[\Pr_{g\in BF_d^m}[P_a(g,x)=f(g,x)]\geq \exp(-2\mu\cdot m)\right]\geq \exp(-\mu\cdot m')\cdot\exp(-\mu\cdot m)/2\geq \exp(-3\mu\cdot m'), \tag{1}$$

where P_a denotes the $(k-1)$-party $(\mu\cdot m')$-bit protocol obtained from P by fixing the first party's message to a (and we are using that $\mu\cdot m'\geq 1$ because $d\geq 1/\mu$). We are now in the position to describe the protocol P'. We define it as a randomized protocol; a deterministic protocol can be obtained by fixing its random choices.

Definition of P'. Let

$$r:=5\cdot\lceil\log(1/\mu)\rceil\cdot d.$$

On input x, the protocol P' first chooses r independent inputs $g_1,g_2,\ldots,g_r$, then runs the protocol P_a on all inputs $(g_1,x),(g_2,x),\ldots,(g_r,x)$. Informally, to compute its j-th output bit, P' uses the output of P_a if one of the g_i's happens to point to j, and otherwise it tosses a coin. More formally, to compute the j-th output bit, P' considers the first pair $u\in[r],w\in[m]$ such that $g_u(w)=j$. If any such u,w exist, then it outputs $P_a(g_u,x)_w$. If no such u,w exist, it outputs a random and uniform bit. it may be the case that P_a gives inconsistent answers on different pairs u,w such that $g_u(w)=j$. While this is addressed in the definition above by using the first such pair, this case of inconsistent pairs u,w is in fact irrelevant to our analysis below. This is because the analysis below gives the desired probability bound just considering the event in which all outputs are correct and hence consistent.)

Complexity of P'. It is easy to verify that the randomized protocol P' can be written as $P'(x)=h\left(P_a(g_1,x),P_a(g_2,x),\ldots,P_a(g_r,x)\right)$, for a function h which depends on the g's. Since P' runs r times the protocol P_a, and P_a uses communication $\mu\cdot m'$, we see that P' uses communication $\mu\cdot m'\cdot r=\mu\cdot m'\cdot 5\cdot\lceil\log(1/\mu)\rceil\cdot d\leq 10\mu\cdot\log(1/\mu)\cdot m'\cdot d$, as required. A deterministic protocol can be obtained by fixing the randomness $\tilde{g}_1=g_1,\tilde{g}_2=g_2,\ldots,\tilde{g}_r=g_r$ so as to maximize the success probability of P'.

Analysis of P'. Call an input $x\in X$ *good* if $\Pr_{g\in BF_d^m}[P_a(g,x)=f(g,x)]\geq\exp(-2\mu\cdot m)$. Our claim is the following.

Claim 4.1. *For every good $x\in X$, $\Pr_{g_1,\ldots,g_r}[P'(x)=f'(x)]\geq\exp(-4\mu\cdot m\cdot r)$.*

[4] Specifically, suppose that $\Pr_{X,Y}[\phi(X,Y)]\geq\gamma$. Let us call x *heavy* if $\Pr_Y[\phi(x,Y)]\geq\gamma'$, and suppose that $\Pr_X[X\text{ is heavy}]=p$. Then we have $\gamma\leq\gamma'(1-p)+p=\gamma'+p(1-\gamma')$, and thus $p>(\gamma-\gamma')/(1-\gamma')$. (The case $\gamma:=\exp(-\mu\cdot m)$, and $\gamma':=\gamma^2$ is used in the proof.)

Before proving this claim, let us see how the theorem follows from it. Note that, by Equation (1), a random $x \in X$ is good with probability at least $\exp(-3\mu \cdot m')$. Combining this with Claim 4.1 we have that

$$\Pr_{x \in X, g_1, \ldots, g_r}[P'(x) = f'(x)] \geq \exp(-3\mu \cdot m' - 4\mu \cdot m \cdot r)$$
$$\geq \exp(-7\mu \cdot m \cdot r) \geq \exp(-70\mu \cdot \log(1/\mu) \cdot m') \quad (2)$$

as required.

Proof of Claim 4.1. The choice of $g_1, \ldots, g_r$ corresponds to the choice of $t := m \cdot r = 5 \cdot \lceil \log(1/\mu) \rceil \cdot m'$ pointers $g_u(w) \in [m']$, where $u \leq r$ and $w \leq m$. Let us call a pointer $g_u(w)$ *good* if P_a correctly computes the corresponding bit, that is, if $P_a(g_u, x)_w = f'(x)_{g_u(w)}$.

Let us pick a parameter

$$\rho \in [4\mu, 5\mu]$$

such that $\rho \cdot d$ is an integer (such a ρ is guaranteed to exist because $d \geq 1/\mu$). Now consider the following two events.

I *All $g_w(w)$'s are good*: For all t pointers $g_u(w)$ we have $P_a(g_u, x)_w = f'(x)_{g_u(w)}$, and

II *Cover a $1-\rho$ fraction*: there are at least $(1-\rho) \cdot m'$ indices $j \in [m']$ for which there is a pointer $g_u(w)$ such that $g_u(w) = j$.

Whenever both events (I) and (II) happen, by definition the protocol will compute correctly a $1-\rho$ fraction of the output bits of f', and guess at random for the other ρ fraction. We prove below that the probability that both events (I) and (II) happen is at least $\exp(-2\mu \cdot m \cdot r)/2$. Also, the probability that the $\rho \cdot m'$ random guesses are all correct is $\exp(-\rho \cdot m')$, and thus we have, for every good x:

$$\begin{aligned} \Pr_{g_1, \ldots, g_r}[P'(x) = f'(x)] &\geq \Pr[\text{(I) and (II)}] \cdot \exp(-\rho \cdot m') \\ &\geq \exp(-2\mu \cdot m \cdot r - 1 - \rho \cdot m') \\ &\geq \exp(-4\mu \cdot m \cdot r), \qquad (\text{since } \rho \leq 5\mu \leq \mu \cdot r/d) \end{aligned}$$

which proves Claim 4.1.[5]

It remains to prove that the probability that both events (I) and (II) happen is at least $\exp(-2\mu \cdot m \cdot r)/2$. First, observe that event (I) happens with probability at least $\exp(-2\mu \cdot m \cdot r)$. This is because x is good and therefore, for every $u \leq r$, with probability at least $\exp(-2\mu \cdot m)$ (over g_u) all pointers $g_u(1), g_u(2), \ldots, g_u(m)$ are good (cf. Equation 1). As these events (for different u's) are independent, the probability bound follows. So we only need to show that event (II) does not happen with probability at most $\exp(-2\mu \cdot m \cdot r)/2$. (We are using that $\Pr[\text{(I) and (II)}] \geq \Pr[\text{(I)}] - \Pr[\text{not (II)}]$.)

Claim 4.2 (Cover a $1-\rho$ fraction w.h.p.)**.** *The probability that there are at most $(1-\rho) \cdot m'$ indices $j \in [m']$ for which there is a pointer $g_u(w)$ such that $g_u(w) = j$ is at most* $\exp(-2\mu \cdot m \cdot r)/2$.

[5] We point out that this idea of guessing at random the missing ρ fraction of bits is borrowed from [BRW]. Just like in that paper, the exponentially small success probability is significant for our setting of parameters. Previously, without using this idea, we had to cope with errors which led to much more complex analysis.

Proof of Claim 4.2. We do a union bound over all subsets of $[m']$ of size $(1-\rho)\cdot m'$. Fix any such set S, and let $1-\rho_1, 1-\rho_2, \ldots, 1-\rho_m$ be the fraction of elements in each block of d indices, i.e., $1-\rho_i := |\{j \in [d\cdot(i-1)+1, d\cdot i] \cap S\}|/d$. The probability that all pointers fall in S is $\prod_{i\le m}(1-\rho_i)^r$. Since the average of the $(1-\rho_i)'s$ is $(1-\rho)$, by the arithmetic mean vs. geometric mean inequality we have that the probability that all pointers fall in S is

$$\prod_{i\le m}(1-\rho_i)^r \le (1-\rho)^{m\cdot r}.$$

By a union bound, the probability that there is a set $S \subseteq [m']$ of size $(1-\rho)\cdot m'$ such that all the pointers fall inside S is at most

$$\begin{aligned}
\binom{m'}{\rho\cdot m'}\cdot(1-\rho)^{m\cdot r} &\le (e/\rho)^{\rho\cdot m'}\cdot\exp(-\rho\cdot m\cdot r)\\
&\le \exp(\log(1/\mu)\cdot\rho\cdot m')\cdot\exp(-\rho\cdot m\cdot r) \qquad (\text{since } \rho\ge 4\mu)\\
&\le \exp(\mu\cdot m\cdot r - 4\mu\cdot m\cdot r) \qquad (\text{since } \rho\in[4\mu,5\mu] \text{ and } r = 5\cdot\lceil\log(1/\mu)\rceil\cdot d)\\
&= \exp(-3\mu\cdot m\cdot r),
\end{aligned}$$

where in the first inequality we use that $\binom{n}{k} \le (e\cdot n/k)^k$ and that $(1-\rho) \le e^{-\rho} \le 2^{-\rho} =: \exp(-\rho)$. □

□

Remark 4.3. *We thank Oded Regev for pointing out an idea from the paper [BRW] which led to a considerable simplifcation of the proof of our main party-reduction theorem. Our previous proof (unpublished) gave up on recovering all the m' bits of f', and only aimed to recover a $1-\rho$ fraction. This parameter ρ then appeared in the induction hypothesis leading to a more cumbersome analysis. Guessing at random the ρ fraction of the bits of f' on which we have no information allows us to have the invariant that we compute all the output bits of f', and leads to a simplified analysis.*

5 Consequences of our lower bound for pointer jumping

In this section we restate and prove some consequences of our lower bound for pointer jumping. We start with the consequences of our lower bound for general protocols with bounded number of rounds: the round hierarchy Theorem 1.1 and the separation of nondeterminism from determinism, Theorem 1.2. Then we prove a separation between the power of different orders in the one-way model, Corollary 1.4. Finally, we give our lower bound for the disjointness function, Corollary 1.5. All these results follow *easily* from our main lower bound for pointer jumping, Theorem 3.2.

We start with the consequences regarding general protocols with a bounded number of rounds. The following corollary combines Theorems 1.1 and 1.2 from the Introduction.

Corollary 5.1 (Separations for a bounded number of rounds)**.** *For every fixed constants r and k there is an explicit function $f : (\{0,1\}^n)^k \to \{0,1\}$ that requires $n^{\Omega(1)}$ communication for k-party r-rounds protocols, but that can be computed with $O(\log n)$ communication by both*

1. *k-party r'-rounds protocols, whenever $r'(k-1) \ge r\cdot k$, and*

2. nondeterministic k-party 2-rounds protocols.

Proof. Let $k' := k \cdot r = O(1)$, and consider the pointer jumping function $TPJ_{k'} : \{0,1\}^n \to \{0,1\}$. Let us think of the input pointers to this function as arranged in r blocks of k layers each, where the first block cointains the first k layers, the second block the layers between $k+1$ and $2 \cdot k$ and so on. Now consider the following partition of the inputs to the k parties: Player i knows all pointers except those leaving the i-th layer in each block. We claim that this function, under this partition of the inputs, gives the separation.

Lower bound: The lower bound follows from our results in the one-way model (Theorem 3.2). Specifically, we use the fact that any r-round k-party protocol for the $TPJ_{k'}$ function above can be simulated by a one-way k'-party protocol, under the partition of the inputs described above (recall that $k' = k \cdot r$). The simulation is natural and works as follows. At any round j, if the i-party speaks in the original r-round protocol, we simulate this by letting the $((j-1) \cdot k + i)$-party speak in the one-way model (i.e., the i-th party in the j-th block). Note that, by our choice of the partition of the inputs, this $((j-1) \cdot k + i)$-party in the one-way protocol has access to all the information that the i-party had access to in the original protocol. This shows that the simulation can indeed be carried through and proves the lower bound.

Upper bound (1): Recall that evaluating the function amounts to following k' pointers, i.e. outputting the node reached after following the pointers. Since party i knows all pointers except those in the i-th layer in each block, it is not hard to see that in 1 round the parties can always follow $k-1$ pointers. Thus, to compute the function they only need r' rounds for $r'(k-1) \geq r \cdot k$. Since at each round they only need communicate the node reached up to that point, and this takes $O(\log n)$ bits, the upper bound is proved.

Upper bound (2): Using nondeterminism, the first party guesses the piece of the input on its forehead, and announces the whole path in the graph from the starting node to a leaf. Note that all the steps in this path are correct except possibly those corresponding to the first layer in each block, which the first party does not see but guessed. As the graph has $k \cdot r = O(1)$ layers, this takes $O(\log n)$ communication. The second party, seeing the first party's input and the path, can verify the correctness of the steps corresponding to the first layer in each block, and thus check if the leaf announced by the first party is the correct one. □

The function witnessing the separation between nondeterminism and determinism in Theorem 5.1 can in fact be computed with $O(\log n)$ communication by both nondeterministic and co-nondeterministic protocols. By a result of [AUY],[6] such functions can always be computed with polylogarithmic communication by deterministic protocols, if we put no bound on the number of rounds. Thus, our exponential separation in Theorem 5.1 provides a sharp contrast in the case of bounded number of rounds.

Another consequence of our results is the following separation between the power of different orders in which the parties speak.

Corollary 1.4 (Order separation)**.** *For any fixed constant k and any order π for the k parties to speak, there is an explicit function $f : (\{0,1\}^n)^k \to \{0,1\}$ that requires $n^{\Omega(1)}$ communication if the parties speak in the order π, but can be computed with $O(\log n)$ communication if the parties speak in any other order.*

[6] Although [AUY] only considers 2-party protocols, their result extends to any k.

Proof. Without loss of generality, let us assume that $\pi_1 = 1, 2, \ldots, k$. Consider the k-party pointer jumping function $TPJ_k : \{0,1\}^n \to \{0,1\}$; by Theorem 3.2, its one-way communication complexity is $\left(n^{1/(k-1)}\right)/k^{O(k)}$ if the parties speak in the order $1, 2, \ldots, k$. For the upper bound, fix a different order, and suppose that party $j > i$ speaks before party i. Then party j can announce the node reached after following the first j pointers, which takes $O(\log n)$ bits. Later, party i can annouce the value of the function. This is because it knows the pointers in every layer except the i-th, and thus in particular the pointers in the layers after the j-th, since $j > i$. □

In the rest of this section we explain how we obtain a lower for the disjointness function.

Definition 5.2 (Disjointness function)**.** *For integers k, n the disjointness function is $DISJ_{k,n} : (\{0,1\}^n)^k \to \{0,1\}$ where $DISJ(x_1, x_2, \ldots, x_k) = 0$ if and only if there is $j \in [n]$ such that $(x_i)_j = 1$ for every $i \in [k]$, where $(x_i)_j$ denotes the j-th bit of the string $x_i \in \{0,1\}^n$.*

Note that $DISJ(x_1, x_2, \ldots, x_k) = 1$ if and only if the k sets encoded by the strings x_i are disjoint.

Corollary 5.3. *There is a universal constant $c > 0$ such that the following is true. For every $k, d \geq 1$, and for $\eta_k := (k+1)^{-c \cdot k}$, there is a distribution D over the inputs of $DISJ_{k,n}$ such that there is no one-way k-party protocol P achieving*

$$\Pr_{x \in D}\left[P(x) = DISJ_{k,n}(x)\right] \geq \exp(-\eta_k)$$

and using communication at most $\eta_k \cdot d$.

In particular, every one-way k-party protocol for the disjointness function $DISJ_{k,n} : (\{0,1\}^n)^k \to \{0,1\}$ must use communication at least $\left(n^{1/(k-1)}\right)/k^{O(k)}$.

Proof. We reduce an instance of the pointer jumping function $TPJ_k^{1,d}$ to an instance of the disjointness function $DISJ_{k,n}$ where $n = d^{k-1}$. The corollary then follows from Theorem 3.2. We now explain this reduction, which amounts to a simple block-wise encoding of the truth-tables of the functions g's. Specifically, given a $TPJ_k^{1,d}$ instance $g_1, g_2, \ldots, g_k$ we construct the corresponding $DISJ_{k,n}$ instance $x_1, x_2, \ldots, x_k$ as follows. The input x_k equals the truth-table of the binary function g_k mapping $[d^{k-1}]$ to $\{0,1\}$. To specify x_i for $i < k$, let us think of the integers up to $n = d^{k-1}$ written in d-ary notation, and of g_i as a map from d-ary integers of length $i-1$ into d-ary integers of length i, i.e. $g_i : [d]^{i-1} \to [d]^i$. Now, we set to 1 exactly those bits of x_i whose index is of the form $g_i(a) \circ b$, $a \in [d]^{i-1}, b \in [d]^{k-i}$. For example, for x_1 we set to 1 all bits whose index has the most significant digit equal to $g_1(1)$. It can be verified that for such x_i's we have $TPJ_k^{1,d}(g_1, g_2, \ldots, g_k) = 1 - DISJ_{k,n}(x_1, x_2, \ldots, x_k)$. □

6 Streaming models and branching programs

In this section we discuss some consequences of our lower bounds for *streaming models* of computation. Following [AMS], we consider the problem of approximating, with limited space, the *frequency moments* of a sequence $A = (a_1, \ldots, a_m)$ of m elements $a_i \in \{1, \ldots, n\}$ (jumping ahead, our current results only apply to the frequency moment F_∞, defined below). The celebrated paper [AMS] establishes several bounds on approximating various moments in the model where we have one-way access to the sequence. Already in [AMS], it was considered an extension of the one-way

model where one allows for *several* one-way passes on the input sequence, but in the *same* order. This model was also studied by some of the many papers that followed [AMS], for example in the paper [BYJKS] which also gives sharper space bounds.

Here, we consider a further extension of the one-way streaming model, where we are allowed to access the sequence in *any* order, insofar as we read at most $c \cdot n$ inputs (the order is fixed and independent of the actual sequence). We can think, for example, of approximating the moments of a matrix of n elements, where we first access them in row order, and then in column order. This overlap of information is reminiscent of the multiparty model, and in fact using our results we are able to prove a space lower bound for approximating the moment F_∞, which is simply the maximum over $i \in \{1, \ldots, n\}$ of the number of times that element i appears in the sequence (cf. [AMS]); for example, $F_\infty = 1$ if all elements in the sequence are distinct.

Corollary 6.1. *For every c there is a constant $\epsilon > 0$ such that the following holds. Let M be a randomized algorithm that uses space s and, given a sequence A of n elements in $\{1, \ldots, n\}$, reads $c \cdot n$ elements in an arbitrary but fixed order (independent of the sequence). Suppose that M outputs with probability $1 - \epsilon$ a number F^*_∞ such that $|F^*_\infty - F_\infty| \leq \epsilon \cdot F_\infty$, then M must use space $s \geq n^\epsilon$.*

We take a slightly indirect route to prove Corollary 6.1. We proceed by introducing the model of *branching programs* and note that it is equivalent to the streaming model in Corollary 6.1; then we infer a lower bound for branching programs from our results. We choose to present things this way because the model of branching programs is interesting in its own right.

Let us start by briefly recalling the definition of a branching program. Because of the application to streaming models, we work with branching programs over n inputs from a non-necessarily Boolean domain D.

Definition 6.2 (Branching program)**.** *A* branching program *on n inputs from a domain D is a directed, acyclic graph whose internal nodes are labeled with queries of the form "input i?", whose edges are labeled with $|D|$ possible query answers, and whose leaves are labeled with real numbers. A branching program computes a function from D^n to the reals in the natural way, by following the path from the root to a leaf. The* length *of a branching program is the length of a longest path from the root to a leaf, and the* width *is the maximum, over all d, of the number of nodes at distance d from the root. A branching program is* oblivious *if any path from the root to a leaf queries the same inputs in the same order.*

We observe that the streaming algorithm M in Corollary 6.1 can be implemented by an oblivious branching program of length $c \cdot n$ and width 2^s (with inputs over the domain $\{1, \ldots, n\}$).

The next lemma states that branching programs can be simulated by multiparty protocols. This simulation was known and indeed motivated the introduction of the multiparty model [CFL]. We make the straightforward observation that the simulation also holds for one-way protocols, which allows us to use the lower bounds in this paper to obtain branching program lower bounds.

Proposition 6.3 ([CFL]; Simulating branching programs by multiparty protocols)**.** *Let $p : D^n \to \Re$ be an oblivious branching program of width w and length $c \cdot n$. For $k = 2c$ and a constant α that depends on c only, there is a partition $H_0, H_1, \ldots, H_k$ of $[n]$ where each set $H_i, i \geq 1$, has size at least $\alpha \cdot n$, and the branching program p can be simulated by a one-way k-party protocol exchanging $(k-1) \cdot \log w + 1$ bits of communication, where party i has the inputs indexed by H_i on his forehead (and the inputs indexed by H_0 are seen by everybody).*

Proof sketch. Divide up the sequence of $c \cdot n$ indices accessed by the program in k blocks of $n/2$ elements each. Start by setting H_i, $i \geq 1$, equal to the complement of indices in the i-th block (clearly this complement has size at least $n/2$). To make the H_i's disjoint, apply iteratively the marriage theorem (a.k.a. Hall's theorem). (Alternatively, assign each index to a random set among those that contain it, and argue that with high probability the resulting sets will be large.) Finally, set $H_0 := [n] - \cup_{i \geq 1} H_i$. □

We are now in the position to present the proof of Corollary 6.1.

Proof of Corollary 6.1. The algorithm M can be seen as an oblivious branching program of length $c \cdot n$ and width 2^s (with inputs over the domain $\{1, \ldots, n\}$). By Proposition 6.3, there is a partition $H_0, H_1, \ldots, H_k$ of $[n]$ where $k = 2c$ and each set $H_i, i \geq 1$ has size at least $\alpha \cdot n$, and the branching program can be simulated by a one-way k-party protocol using communication $(k-1) \cdot s + 1$ (on that partition). Similarly to the approach in [AMS], we now argue that such a protocol can be used to compute the k-party disjointness function $DISJ_{k,\alpha \cdot n} : (\{0,1\}^{\alpha \cdot n})^k \to \{0,1\}$. Given an input $x_1, \ldots, x_k \subseteq [\alpha \cdot n]$ of the disjointness function, we construct the sequence as follows. For every i, we set $|x_i|$ of the elements indexed by H_i to the elements in the input set $x_i \subseteq [\alpha \cdot n]$, in arbitrary order. The remaining $|H_0| + \sum_i |H_i| - |x_i|$ elements of the sequence can be, say, assigned to distinct elements from $\{\alpha \cdot n + 1, \ldots, n\}$. It is easy to see that the moment F_∞ of this sequence is at most $k-1$ if $DISJ_{k,\alpha \cdot n}(x_1, \ldots, x_k) = 1$, and otherwise it is k. Thus, for a sufficiently small constant ϵ depending on k, approximating F_∞ lets us compute disjointness, and by Corollary 5.3 we conclude that $s \geq n^\epsilon$. □

Finally, we mention that we do not know how to get hardness of approximation results for other moments in our model.

7 Open problems

Besides the obvious open problem of establishing lower bounds for pointer jumping or disjointness in the general model, our work raises some other questions. One question, in proof complexity, is whether a size lower bound for tree-like threshold systems already follows from our one-way lower bound (cf. Section 1.3 and [BPS]). This seems to require a "better than known" simulation of such proof systems by multi-party protocols, specifically a simulation that uses only few rounds. Another question is how tight is our one-way lower bound for pointer jumping if k is not constant. A small step in this direction would be understanding whether one can improve our main Theorem 3.3 so as to have η' linear in η, as opposed to quasi-linear. This would allow to establish lower bounds for pointer jumping of the form $\left(n^{1/(k-1)}\right)/2^{O(k)}$, as opposed to $\left(n^{1/(k-1)}\right)/k^{O(k)}$ in our results.

Acknowledgments. We thank Noam Nisan for helpful conversations, and in particular for suggesting the application to round separation. We are grateful to Oded Regev for pointing out an idea from the paper [BRW] which led to a considerable simplification of the proof of our main party-reduction theorem (see Remark 4.3). Finally, we would like to thank Ronald de Wolf and the anonymous referees for helpful comments.

References

[AUY] A. V. Aho, J. D. Ullman, and M. Yannakakis. On notions of information transfer in VLSI circuits. In *Proceedings of the 15th annual ACM symposium on Theory of computing*, pages 133–139, New York, NY, USA, 1983. ACM Press. 12

[AMS] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. System Sci.*, 58(1, part 2):137–147, 1999. Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996). 2, 13, 14, 15

[BHK] L. Babai, T. P. Hayes, and P. G. Kimmel. The cost of the missing bit: communication complexity with help. *Combinatorica*, 21(4):455–488, 2001. 1, 2, 3, 6

[BNS] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. System Sci.*, 45(2):204–232, 1992. Twenty-first Symposium on the Theory of Computing (Seattle, WA, 1989). 2

[BYJKS] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. System Sci.*, 68(4):702–732, 2004. 2, 14

[BIP] P. Beame, R. Impagliazzo, and T. Pitassi. Improved depth lower bounds for small distance connectivity. *Comput. Complexity*, 7(4):325–345, 1998. 2

[BPS] P. Beame, T. Pitassi, and N. Segerlind. Lower bounds for Lovász-Schrijver systems and beyond follow from multiparty communication complexity. In *Automata, languages and programming*, volume 3580 of *Lecture Notes in Comput. Sci.*, pages 1176–1188. Springer, Berlin, 2005. 3, 15

[BPSW] P. Beame, T. Pitassi, N. Segerlind, and A. Wigderson. A Direct Sum Theorem for Corruption and the Multiparty NOF Communication Complexity of Set Disjointness. In *Proceedings of the 20th Annual Conference on Computational Complexity*, pages 52–66. IEEE, June 12–15 2005. 1, 3

[BRW] A. Ben-Aroya, O. Regev, and R. d. Wolf. A Hypercontractive Inequality for Matrix-Valued Functions with Applications to Quantum Computing, 2007. arXiv:0705.3806v1. 7, 10, 11, 15

[Cha] A. Chakrabarti. Lower Bounds for Multi-Player Pointer Jumping. In *Proceedings of the 22nd Annual Conference on Computational Complexity*. IEEE, June 13–16 2007. 1, 2

[CFL] A. K. Chandra, M. L. Furst, and R. J. Lipton. Multi-party protocols. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, Boston, Massachusetts, 25–27 Apr. 1983. 1, 14

[CT] F. R. K. Chung and P. Tetali. Communication complexity and quasi randomness. *SIAM J. Discrete Math.*, 6(1):110–123, 1993. 2

[DJS] C. Damm, S. Jukna, and J. Sgall. Some bounds on multiparty communication complexity of pointer jumping. *Comput. Complexity*, 7(2):109–127, 1998. 1, 2, 4

[KN] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, Cambridge, 1997. 1, 6

[NBY] N. Nisan and Z. Bar-Yossef. Pointer jumping requires concurrent read. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 549–558 (electronic), El Paso, Texas, 4–6 May 1997. 2

[NW] N. Nisan and A. Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, 1993. 1, 2, 4

[PS] C. H. Papadimitriou and M. Sipser. Communication complexity. *J. Comput. System Sci.*, 28(2):260–269, 1984. 1, 2

[Raz] R. Raz. The BNS-Chung criterion for multi-party communication complexity. *Comput. Complexity*, 9(2):113–122, 2000. 2

[RW] A. Razborov and A. Wigderson. $n^{\Omega(\log n)}$ lower bounds on the size of depth-3 threshold circuits with AND gates at the bottom. *Inform. Process. Lett.*, 45(6):303–307, 1993. 3

[Vio] E. Viola. Pseudorandom Bits for Constant-Depth Circuits with Few Arbitrary Symmetric Gates. *SIAM Journal on Computing*, 36(5):1387–1403, 2007. 3

[VW] E. Viola and A. Wigderson. Norms, XOR lemmas, and lower bounds for GF(2) polynomials and multiparty protocols. In *Proceedings of the 22nd Annual Conference on Computational Complexity*. IEEE, June 13–16 2007. 2

[Yao] A. C.-C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th annual ACM symposium on Theory of computing*, pages 209–213, New York, NY, USA, 1979. ACM Press. 1

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/