Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 24 (2008)

On the Value of Multiple Read/Write Streams for Approximating Frequency Moments

Paul Beame*
Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
beame@cs.washington.edu

Dang-Trinh Huynh-Ngoc†
Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
trinh@cs.washington.edu

April 30, 2008

Abstract

Recently, an extension of the standard data stream model has been introduced in which an algorithm can create and manipulate multiple read/write streams in addition to its input data stream. Like the data stream model, the most important parameter for this model is the amount of internal memory used by such an algorithm. The other key parameters are the number of streams the algorithm uses and the number of passes it makes on these streams. We consider how the addition of these multiple read/write streams impacts the ability of algorithms to approximate the frequency moments of the input stream.

We show that any randomized read/write stream algorithm with a fixed number of streams and a sublogarithmic number of passes that produces a constant factor approximation of the k-th frequency moment F_k of an input sequence of length of at most N from $\{1, \ldots, N\}$ requires space $\Omega(N^{1-4/k-\delta})$ for any $\delta > 0$. For comparison, it is known that with a single read-only data stream there is a randomized constant-factor approximation for F_k using $\tilde{O}(N^{1-2/k})$ space and that there is a deterministic algorithm computing F_k exactly using 3 read/write streams, $O(\log N)$ passes, and $O(\log N)$ space. Therefore, although the ability to manipulate multiple read/write streams can add substantial power to the data stream model, with a sub-logarithmic number of passes this does not significantly improve the ability to approximate higher frequency moments efficiently. Our lower bounds also apply to $(1+\epsilon)$-approximations of F_k for $\epsilon \geq 1/N$.

1 Introduction

The development of efficient on-line algorithms for computing various statistics on streams of data has been a remarkable success for both theory and practice. The main model has been the data stream model in which algorithms with limited storage access the input data in one pass as it streams by. This model is natural for representing many problems in monitoring web and transactional traffic.

The potential of the data stream model became clear with the groundbreaking paper by Alon, Matias, and Szegedy [AMS99] which showed that one-pass randomized algorithms with surprisingly

*Research supported by NSF grant CCF-0514870
†Research supported by a Vietnam Education Foundation Fellowship

ISSN 1433-8092

small space requirements can approximately determine the *frequency moments* of data streams. The k-th frequency moment, F_k, is the sum of the k-th powers of the frequencies with which elements occur in a data stream. F_1 is simply the length of the data stream; F_0 is the number of distinct elements in the stream; if the stream represents keys of a database relation then F_2 represents the size of the self-join on that key. The methods in [AMS99] also yielded efficient randomized algorithms for approximating F_∞^*, the largest frequency of any element in the stream. These results have been extended and improved to apply to many other problems, including approximating arbitrary join sizes, computation of ℓ_p differences between data streams, and computations over sliding windows (see surveys [Mut06, BBD$^+$02]). The best one-pass algorithms for frequency moments approximate F_k within a $(1+\epsilon)$ factor on streams of length N using $\tilde{O}(N^{1-2/k})$ space [IW05, BGKS06].

Along with designing algorithms for approximating F_k, Alon, Matias, and Szegedy showed that their algorithms were not far from optimal in the one-pass model; in particular, they showed that F_k requires $\Omega(N^{1-5/k})$ space to approximate by randomized one-pass algorithms. They derived their lower bounds by extending bounds [Raz92] for the randomized 2-party communication complexity for a promise version of the set disjointness problem from 2 to p players, where each of the p players has access to its own private portion of the input. (The model is known as the *p-party number-in-hand communication game*.) A series of papers [SS02, BYJKS04, CKS03] has improved the space lower bound to an essentially optimal $\tilde{\Omega}(N^{1-2/k})$ by improving the lower bound for the promise disjointness problem for p-party randomized number-in-hand communication games; thus F_k for $k > 2$ requires polynomial space in the data stream model[1].

However, as Grohe and Schweikardt [GS05] observed, in many natural situations for which the data stream model has been studied, the computation also has access to auxiliary external memory for storing intermediate results. In this situation, the lower bounds for the data stream model no longer apply. This motivated Grohe and Schweikardt to introduce a model, termed the *read/write streams model* in [BJR07], to capture this additional capability. In the read/write streams model, in addition to the input data stream, the computation can manipulate multiple sequentially-accessed read/write streams.

As noted in [GS05], the read/write streams model is substantially more powerful than the ordinary data stream model since read/write stream algorithms can sort lists of size N with $O(\log N)$ passes and space using 3 streams and hence compute any F_k exactly using only $O(\log N)$ passes, $O(\log N)$ space, and 3 streams. Unfortunately, given the large values of N involved, $\Theta(\log N)$ passes through the data is a very large cost. For sorting, lower bounds given in [GS05, GHS06] show that such small space read/write stream algorithms are not possible using fewer passes; moreover, [GHS06, BJR07] show lower bounds for the related problems of determining whether two sets are equal and of determining whether or not the input stream consists of distinct elements.

However, these bounds say very little about the problem of approximating frequency moments, which has much less stringent requirements than the above problems. Can read/write stream algorithms approximate larger frequency moments more efficiently than single pass algorithms can? It seems plausible that read/write stream algorithms might be able to compute F_k efficiently for larger k than is possible for data stream algorithms.

We show that the ability to augment the data stream model with computations using multiple read/write streams does not produce significant additional efficiency in approximating frequency

[1]The $\Omega(1/\epsilon^2)$ dependence of the space on the error in the approximation has also been shown to be optimal for one-pass algorithms [IW03, Woo04] though by considering a different two-party communication complexity problem – approximate Hamming distance

moments. In particular, any randomized read/write stream algorithm with a fixed number of streams and $o(\log N)$ passes that approximates the k-th frequency moment F_k of an input sequence of length of at most N from $\{1, \ldots, N\}$ within a constant factor requires space $\Omega(N^{1-4/k-\delta})$ for any $\delta > 0$. This lower bound is very similar to the upper bound even for standard single pass read-only data streams (and is larger than the original lower bound in [AMS99] for ordinary data streams).

The major difficulty in developing lower bounds for the read/write streams model, in contrast to the data streams model, is that an easy reduction from number-in-hand multiparty communication complexity game breaks down. This easy reduction fails in the read/write stream model because of the repeated access to the data and the ability of different parts of the computation to communicate with each other via the multiple read/write streams. Nowhere is this more evident than in the complexity of the p-party disjointness promise problem in the read/write streams model. This problem, which is the basis for the lower bounds for approximating frequency moments in the data streams model, can be easily solved by read/write stream algorithms using only 3 passes, 2 streams and $O(\log N)$ space.

The amount of data written on the streams also prevents the use of traditional time-space tradeoff lower bound methods, which are the other obvious tools to consider. As a result, previous work on lower bounds in the read/write streams model has developed special-purpose combinatorial methods designed especially for the model.

Grohe, Hernich, and Schweikardt [GS05, GHS06] found certain structural properties of the executions of read/write stream algorithms, their *skeletons*, and used these skeletons together with cut-and-paste arguments to show the existence of certain combinatorial rectangles on which the algorithms' answers must be constant. As a result they were able to show that deterministic and randomized algorithms with $o(\log N)$ passes and one-sided error are unable to sort efficiently or to determine whether or not two input sets are equal. Based on these bounds they also derived lower bounds for one-sided error randomized algorithms for a number of other problems.

Beame, Jayram, and Rudra [BJR07] used the skeleton characterization and the existence of combinatorial rectangles as shown in [GS05, GHS06], together with additional combinatorial reasoning to show how standard properties used for lower bounding randomized two-party communication complexity, namely discrepancy and corruption over rectangles, could be used to derive lower bounds for randomized read/write streams with two-sided error. Using this approach they gave some general methods for obtaining lower bounds for two-sided error randomized read/write stream algorithms. In particular they showed that with $o(\log N/\log\log N)$ passes and $O(N^{1-\delta})$ space, randomized read/write stream algorithms with two-sided error cannot determine whether or not two input sets are disjoint. This yielded a number of other lower bounds, including an $\Omega(N^{1-\delta})$ lower bound on the space for computing a 2-approximation of F_∞^* with $o(\log N/\log\log N)$ passes and a similar lower bound for exact computation of F_0. However, the methods of [BJR07] do not yield lower bounds on the approximate computation of frequency moments F_k for any other values of k. In particular it was consistent with their work that read/write algorithms could compute constant factor approximations to any such F_k using $o(\log N)$ passes, $O(\log N)$ space, and only 2 streams.

We take a quite different approach to lower bounds in the read/write streams model from those in [GS05, GHS06, BJR07]. Despite the failure of the standard reduction, we are able to characterize read/write stream algorithms by developing a novel direct simulation of read/write stream algorithms by p-party communication protocols. Though quite different in the overall structure of the argument, this reduction does make use of a simplified variant of the skeletons

defined in [GS05, GHS06]. Our method may have many other applications.

For the specific case of approximating frequency moments we derive our lower bounds by applying our simulation to a blocked and permuted version of the promise p-party disjointness problem (with p depending on N and k). The problem is a generalization of one considered in [BJR07] extended to the case of p sets. This allows us to obtain the $\Omega(N^{1-4/k-\delta})$ space lower bounds for computing F_k using a sublogarithmic number of passes and a constant number of streams.

Although this nearly matches the best lower bounds for the data stream model, there is still a gap between our read/write streams lower bounds and the data stream upper bounds because our lower bounds are limited by the relationship between the number of blocks and the number of sets in the permuted disjointness problem we consider. We also show that modifying this relationship cannot improve the lower bound for constant factor approximations for $k < 3.5$ by showing that for any relationship that would give a better bound for these values of k there is a deterministic read/write stream algorithm with three passes, two streams and only $O(\log N)$ space that can compute the value of the blocked and permuted p-party disjointness problem. To derive this algorithm we show a novel property on the lengths of common subsequences in sets of permutations that is interesting in its own right.

2 Preliminaries

In the read/write streams model, the streams are represented as t read/write Turing machine tapes. The input stream is given as the contents of the first such tape; the other streams/tapes are used for working storage. Passes through the data in a stream correspond to reversals on the corresponding Turing machine tape; the number of passes is one more than the number of reversals. The internal memory of read/write streams allows random access.

The three resource parameters that are important to a read/write stream algorithm A are (1) the number of external read/write tapes t that A uses, (2) the maximum space s that A uses, and (3) the maximum number of reversals r made by A on all the external tapes.

Since we will primarily focus on lower bounds, we define a nonuniform version of the read/write stream model since lower bounds for this model are more general than those that only apply to the uniform case. Fix an input alphabet Σ and tape alphabet Γ. An *(r, s, t)-read/write stream algorithm* A on Σ^N is an automaton with 2^s states with one read/write head on each of t tapes. It begins with its input $\mathbf{v} \in \Sigma^N$ on the first tape and the remaining tapes blank. In each step, based on the current state and currently scanned symbols, one of its heads writes a new symbol from Γ in its currently scanned tape cell and moves one cell left or right. On any input $\mathbf{v} \in \Sigma^N$ it reverses the direction of movement of its heads at most r times before it halts.

For functions $r, s : \mathbb{N} \to \mathbb{N}$ and $t \in \mathbb{N}$, a (nonuniform) $(r(\cdot), s(\cdot), t)$-read/write stream algorithm on Σ^* is a family of algorithms $\{\mathsf{A}_N\}_{N\in\mathbb{N}}$ where for each N, A_N is an $(r(N), s(N), t)$-read/write stream algorithm and all A_N have the same input and tape alphabets. Randomized and nondeterministic algorithms are defined analogously.

For integer $m \geq 1$ denote $\{1, \ldots, m\}$ by $[m]$. For a set T, we write $\mathcal{S}_T$ to denote the set of strings of length $|T|$ that are permutations of T. A string s of length $|s|$ is said to be the *interleaving* of another set of strings $\{s_1, \ldots, s_t\}$ if there is a partition of $\{1, \ldots, |s|\}$ into t subsets $\{q_1, \ldots, q_t\}$ such that for every $1 \leq i \leq t$, $s_{|q_i} = s_i$, where $s_{|q_i}$ denotes the string obtained from s projected on coordinates in q_i only, and for every $j \in q_i$, the j-th entry of s is said to be *assigned* to s_i.

For any $m \geq 1$ and permutation ϕ of $[m]$, define the *sortedness* of ϕ, denoted by $\mathsf{sortedness}(\phi)$,

to be the length of the longest monotone subsequence of $(\phi(1), \ldots, \phi(m))$. For any $m, p \geq 1$ and a sequence $\Phi = (\phi_1, \phi_2, \ldots, \phi_p)$ of permutations of $[m]$, define the *relative sortedness* of Φ, denoted by $\mathsf{relsorted}(\Phi)$, to be $\max_{i \neq j \in [p]}(\mathsf{sortedness}(\phi_i \phi_j^{-1}))$.

Lemma 2.1 (cf. [Ste97]; Lemma 1.4.1)**.** *Let ϕ be a randomly and uniformly chosen permutation of $[m]$, then* $\Pr[\mathsf{sortedness}(\phi) \geq 2e\sqrt{m}] < 2\exp(-2e\sqrt{m})$.

Corollary 2.2. *If $p = m^c$ for some constant $c > 0$ and m is sufficiently large there exists a sequence $\Phi = (\phi_1, \phi_2, \ldots, \phi_p)$ of permutations of $[m]$ such that* $\mathsf{relsorted}(\Phi) < 2e\sqrt{m}$.

For a given p polynomial in m, we let Φ^* denote a sequence that is guaranteed to exist by Corollary 2.2.

The *k-th frequency moment* F_k of a sequence $a_1, \ldots, a_n \in [m]$ is $\sum_{j \in [m]} f_j^k$ where $f_j = \#\{i \mid a_i = j\}$. We will typically consider the problem when $m = n$. Also write F_∞^* for $\max_{j \in [m]} f_j$

For $2 \leq p < n$, we define the promise problem $\textsc{pDisj}_{n,p} : \{0,1\}^{np} \mapsto \{0,1\}$ as follows: For $x_1, \ldots, x_p \in \{0,1\}^n$ we interpret each x_i as the characteristic function of a subset of $[n]$. If these subsets are pair-wise disjoint then $\textsc{pDisj}(x_1, \ldots, x_p) = 0$; if there is a unique element $a \in [n]$ such that $x_i \cap x_j = a$ for all $i, j \in [p]$ then $\textsc{pDisj}(x_1, \ldots, x_p) = 1$; otherwise, the function is undefined.

We use the usual definition of p-party number-in-hand communication complexity. A series of communication complexity lower bounds [AMS99, SS02, BYJKS04, CKS03] for $\textsc{pDisj}_{n,p}$ in this number-in-hand model has resulted in essentially optimal lower bounds for computing frequency moments in the data stream model, even allowing multiple passes on the input stream. The strongest of these bounds [CKS03] shows that any p-party public-coin randomized number-in-hand communication protocol for $\textsc{pDisj}_{n,p}$ must have complexity at least $\Omega(\frac{n}{p \log p})$. (The bound is an even stronger $\Omega(n/p)$ for one-way communication.)

However, as noted in the introduction, for any n and p, there is a simple $(2, \log_2 n + O(1), 2)$ read/write stream algorithm for computing $\textsc{pDisj}_{n,p}$: Simply copy the bits of x_1 from the input tape to the second tape and, after traversing the bits of x_2 in the input tape, reverse the heads on both tapes and compare x_2 on the first tape bit-by-bit to x_1 on the second tape. We therefore will need a modified function in order to obtain our lower bounds for approximating frequency moments.

Let $N > p \geq 2$ and let $\Pi = (\pi_1, \ldots, \pi_p)$ be a sequence of permutations on $[N]$. We define the promise problem $\textsc{pDisj}^{\Pi}_{N,p} : \{0,1\}^{Np} \mapsto \{0,1\}$ by $\textsc{pDisj}^{\Pi}_{N,p}(y_1, \ldots, y_p) = \textsc{pDisj}_{N,p}(x_1, \ldots, x_p)$ where the j-th bit of x_i is the $\pi_i^{-1}(j)$-th bit of y_i. The relationship between x_i and y_i is equivalent to requiring that $y_{ij} = x_{i\pi_i(j)}$.

We first observe that the same reduction idea given by [AMS99] yields lower bounds for F_k given lower bounds for $\textsc{pDisj}^{\Pi}_{N,p}$ for suitable choices of p.

Lemma 2.3. *Let $N > 1$ and $1/2 > \delta \geq 0$.*

(a) Let $k > 1$, $1 \geq \epsilon > 4^{1/(k-1)}/N$, and $p \geq (4\epsilon N)^{1/k}$. If there is a randomized (r, s, t)-read/write stream algorithm that outputs an approximation of F_k within a $(1+\epsilon)$-factor on a sequence of at most N elements in the range $[N]$ with probability at least $1 - \delta$ then there is a randomized $(r + 2, s + O(\log N), \max(t, 2))$-read/write stream algorithm with error at most δ that solves $\textsc{pDisj}^{\Pi}_{N,p}$ for any Π.

(b) Let $k < 1$, $1/2 > \epsilon \geq 0$, and let $p - p^k \geq 2\epsilon N$. If there is a randomized (r, s, t)-read/write stream algorithm that outputs an approximation of F_k within a $(1+\epsilon)$-factor on a sequence of

at most N elements in the range $[N]$ with probability at least $1-\delta$ then there is a randomized $(r+2, s+O(\log N), \max(t,2))$-read/write stream algorithm with error at most δ that solves $\text{pDISJ}^{\Pi}_{N,p}$ for any Π.

Proof. We show part (a) first. Given an input $\mathbf{v}$ to $\text{pDISJ}^{\Pi}_{N,p}$, the read/write stream algorithm reads its bits from left to right to convert it on the second tape to the appropriate input $\mathbf{v}'$ for the F_k problem in the obvious way: Read $\mathbf{v}$ from left to right, for every bit that is as 1 in position $i \in [N]$ in the input, Write i to $\mathbf{v}'$. While doing this also use the state to record the number $N' \leq N$ of 1's in the input. If $N' > N$ then output 1 and halt. Otherwise copy tape 2 to tape 1, erasing tape 2. By the promise, when $\text{pDISJ}^{\Pi}_{N,p} = 0$, we have $F_k(\mathbf{v}') = N' \leq N$; when $\text{pDISJ}^{\Pi}_{N,p} = 1$, we have

$$\begin{aligned} F_k(\mathbf{v}') &= N' - p + p^k \\ &\geq N' + 4\epsilon N - (4\epsilon N)^{1/k} \\ &\geq (1+\epsilon)^2 N' + \epsilon N - (4\epsilon N)^{1/k} \\ &> (1+\epsilon)^2 N' \end{aligned}$$

where the second inequality follows because $\epsilon \leq 1$ and the third follows because $\epsilon > 4^{1/(k-1)}/N$. The algorithm for $\text{pDISJ}^{\Pi}_{N,p}$ will output 1 if the value returned for F_k is greater than $(1+\epsilon)N'$ and output 0 otherwise.

For part (b) which has $k < 1$, we apply the same algorithm as in part (a), except that the input promises yield different implications for the relative values of F_k and therefore the output condition will be different. If $\text{pDISJ}^{\Pi}_{N,p}(\mathbf{v}) = 0$ then $F_k(\mathbf{v}') = N' \leq N$ as before but, since $k < 1$, if $\text{pDISJ}^{\Pi}_{N,p}(\mathbf{v}) = 1$ then $F_k(\mathbf{v}') < N'$. In particular, $F_k(\mathbf{v}') = N' - p + p^k < N'/(1+\epsilon)^2$ if $p - p^k \geq 2\epsilon N$ since $2\epsilon > \frac{2\epsilon+\epsilon^2}{1+2\epsilon+\epsilon^2}$ for $1 \geq \epsilon \geq 0$. The algorithm for $\text{pDISJ}^{\Pi}_{N,p}$ will output 1 if the value returned for F_k is smaller than $N'/(1+\epsilon)$ and output 0 otherwise. □

In our lower bound argument we will find it convenient to work with a special case of $\text{pDISJ}^{\Pi}_{N,p}$ in which the sequence of permutations Π is of a special form. Let $p \geq 2$ and $N = mn$ where m and n are integers. A sequence $\Pi = (\pi_1, \ldots, \pi_p)$ of permutations on $[N]$ has *(monotone) block-size m* if and only if there is a sequence $\Phi = (\phi_1, \ldots, \phi_p)$ of permutations on $[m]$ such that $\pi_i(j) = (\phi_i(j')-1)n + j''$ where $j = (j'-1)n + j''$ with $j'' \in [n]$. That is each permutation π_i permutes blocks of length n in $[N]$ but leaves each block intact. In this case, we write $\text{pDISJ}^{m,\Phi}_{n,p}$ for $\text{pDISJ}^{\Pi}_{N,p}$.

Note that the function $\text{pDISJ}^{m,\Phi}_{n,p}$ can be viewed as the logical $\vee$ of m independent copies of $\text{pDISJ}_{n,p}$ in which the input blocks for the different functions have been permuted by Φ. In particular, using an extension of the notation of [BJR07], we see that $\text{pDISJ}^{m,\Phi}_{n,p} = f^{\vee}_{\Phi}$ where $f = \text{pDISJ}_{n,p}$, and $f^{\vee}_{\Phi}(\mathbf{v}_{11}, \ldots, \mathbf{v}_{1m}, \ldots, \mathbf{v}_{p1}, \ldots, \mathbf{v}_{pm}) = \bigvee_{j \in [m]} f(\mathbf{v}_{1\phi_1(j)}, \ldots, \mathbf{v}_{p\phi_p(j)})$.

3 Information flow in the read/write streams model

In this section we prove several results about the "information flow" in an algorithm's execution in the read/write streams model. These results capture the information flow among the tapes in various stages of the computation. In this section, we only consider deterministic (r, s, t)-read/write stream algorithms.

Information flow in a read/write stream algorithm's execution is captured via an object called a *dependency graph* which shows the dependence of portions of the tapes on a given input string at various stages of a computation. This notion of dependency graphs is adapted from and simpler than the concept of *skeletons* that is used in [GS05, GHS06, BJR07]; although much simpler, it suffices for our purposes.

Recall that the input is a string $\mathbf{v} \in \{0,1\}^*$ which is written on the first tape and that at any step only one of the t heads moves. Assume without loss of generality that the read/write stream algorithm A makes a total of r reversals on input $\mathbf{v}$. The dependency graph corresponding to $\mathbf{v}$, denoted by $\sigma(\mathbf{v})$, has $r+2$ levels – level 0 corresponds to the beginning of the computation and level $r+1$ corresponds the end of the computation. Level k for $1 \leq k \leq r$ encodes the dependency on the input of each of the t tapes immediately before the k-th reversal in the following manner: For $0 \leq k \leq r+1$ there is one node at level k of $\sigma(\mathbf{v})$ for each tape cell that either contained a symbol of input $\mathbf{v}$ or was visited at some time during the computation on input $\mathbf{v}$ before the k-th reversal, or before the end of the computation for $k = r+1$. The nodes at level k are ordered according to their positions on their corresponding cells on the tapes. Because of this we can view nodes of the dependency graph, interchangeably, as tape cells. There are pointers to each node at level k from each of the nodes in level $k-1$ that it *depends* on.

The crucial observation made in [GS05] about read/write stream algorithms is the following: When a symbol is written in a particular cell by the read/write stream algorithm between its $k-1$-st and k-th reversal (i.e, at level k of $\sigma(\mathbf{v})$), what is being written in that cell can only depend on the current state and the t symbols currently being scanned by the read/write heads. However, the values of these t symbols were determined *before* the $k-1$-st reversal. This implies that any cell at level k depends either on t cells in level $k-1$ (when it is overwritten in level k) or only on itself in level $k-1$ (when it is intact in level k). The dependency graph thus consists of a layered directed graph of tape cells of in-degree either t or 1 representing the cell dependencies, where all the edges connect consecutive layers.

For some $b \geq 1$, suppose that $\mathbf{v}$ is partitioned into b blocks of consecutive bits: $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_b)$. For every cell c in $\sigma(\mathbf{v})$, we write the *input dependency* of c as $\widetilde{I}_b(c) \subset \{1, \ldots, b\}$ to denote the set of input blocks that it *depends* on (i.e, the set of i, for $1 \leq i \leq b$, such that there is a path from a cell in $\mathbf{v}_i$ at level 0 to c). We note that the set $\widetilde{I}_b(c)$ depends on how we partition $\mathbf{v}$, which explains the subscript 'b' in the notation. Since in this paper we are only interested in those partitions into equal-length blocks, this notation suffices; moreover, we will sometimes drop the subscript 'b' if it is clear from the context. It is immediate from the definition that for every cell c in level k for $0 \leq k \leq r+1$, $|\widetilde{I}(c)| \leq t^k$.

For any cell c on a tape, we write $r(c)$ and $\ell(c)$ for the cells immediately to its right and to its left, respectively.

Proposition 3.1. *Suppose that an input* $\mathbf{v}$ *is partitioned into* b *blocks:* $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_b)$ *for some* $b \geq 1$. *Let* C *be the set of cells on any one tape at any level* k *in* $\sigma(\mathbf{v})$, *for* $0 \leq k \leq r+1$. *Fixing any* $1 \leq i \leq b$, *let* $S = \{c \in C \mid i \in \widetilde{I}(c) \text{ and } i \notin \widetilde{I}(r(c))\}$ *and* $S' = \{c \in C \mid i \in \widetilde{I}(c) \text{ and } i \notin \widetilde{I}(\ell(c))\}$. *Then* $|S| = |S'|$ *and* $|S| \leq t^k$.

Proof. The first part is obviously true, so we just prove the second part. We proceed by induction on k. For $k = 0$, this is true since the only cell in S is the one at the right boundary of the i-th input.

For the induction step, consider any $k > 0$. At level k, a cell gets its input dependency from at most t cells on t tapes at level $k-1$ that it depends on. Thus, for any cell $c \in S$, there must be

some cell $\tilde{c}$ from level $k-1$ such that c depends on $\tilde{c}$, $i \in \widetilde{I}(\tilde{c})$, and either $r(c)$ depends on $r(\tilde{c})$ and $i \notin \widetilde{I}(r(\tilde{c}))$, or $r(c)$ depends on $\ell(\tilde{c})$ and $i \notin \widetilde{I}(\ell(\tilde{c}))$ (depending on whether c's tape head direction and $\tilde{c}$'s tape head direction are the same or not between the $(k-1)$-st and k-th reversals). By induction, this happens at most $t \cdot t^{k-1} = t^k$ times. □

Consider the dependency graph $\sigma(\mathbf{v})$ associated with an input $\mathbf{v}$. For any level k in $\sigma(\mathbf{v})$ consider the sequence of nodes in $\sigma(\mathbf{v})$ corresponding to one of the tapes. An *input dependency string* C of this tape is any string in $\mathcal{S}_{\widetilde{I}(c_1)} \cdots \mathcal{S}_{\widetilde{I}(c_L)}$ where the cells of the tape are $c_1, \ldots, c_L$ in order. For any cell c_i, a string in $\mathcal{S}_{\widetilde{I}(c_i)}$ that is a substring of C is called a *cell portion* of C associated with c_i.

Proposition 3.2. *Let C be an input dependency string of any one tape at any level k in $\sigma(\mathbf{v})$, for $0 \le k \le r+1$. Then C can be written as the interleaving of at most t^k monotone sequences so that for every such sequence s, there is at most one entry in every cell portion of C that is assigned to s.*

Proof. The general idea is taken from [GHS06]. We proceed by induction on k. We will prove a somewhat stronger inductive claim, namely that the property above also holds for more general strings, namely those strings C in $\bigcup_{a_i \ge 1} \mathcal{S}^{a_1}_{\widetilde{I}(c_1)} \cdots \mathcal{S}^{a_L}_{\widetilde{I}(c_L)}$, which we call the set of *extended dependency sequences* for a tape with cells $c_1, \ldots, c_L$. Note that each of these strings can be partitioned into $\sum_{i=1}^{L} a_i$ cell portions, each of which corresponds to a string in $\mathcal{S}_{\widetilde{I}(c_i)}$ for some $1 \le i \le L$.

For $k = 0$, the only non-empty tape is the input tape and C itself is a monotone sequence. Thus this is true for $k = 0$.

For the induction step, suppose the tape we are considering is the j-th tape, where $1 \le j \le t$. We note that at level k the algorithm overwrites a subset of consecutive cells in the tape and the remaining cells are kept intact. Thus C can be written as $C = C'DC''$, where C' and C'' correspond to those cells that are intact from level $k-1$ and D corresponds to those cells that are overwritten. For each of those former cells, its input dependency is unchanged from level $k-1$, and for each of those latter cells, its input dependency is the union of those of the t cells it depends on. Thus D can be written as the interleaving of t sequences $D_1, \ldots, D_t$, where sequence D_i for $1 \le i \le t$ denotes a substring of an extended input dependency of tape i from level $k-1$. One observation here is that $C'D_jC''$ is also a substring of an extended input dependency string of tape j at level $k-1$. By induction, each of $D_1, \ldots, D_t$ and $C'D_jC''$ can be written as the interleaving of at most t^{k-1} monotone sequences satisfying the requirement. Hence C can be written as the interleaving of at most t^k monotone sequences satisfying the requirement. □

Proposition 3.3. *Suppose that an input $\mathbf{v}$ is partitioned into b blocks $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_b)$ for some $b \ge 1$. For any cell c (at any level) in $\sigma(\mathbf{v})$, let $H(c) = \big\{\{i,j\} \mid 1 \le i \ne j \le b \text{ and } i,j \in \widetilde{I}(c)\big\}$. Then $|\cup_{c \in \sigma(\mathbf{v})} H(c)| \le t^{3r+4}b$.*

Proof. Note that for any two input blocks $\mathbf{v}_i$ and $\mathbf{v}_j$, if $i,j \in \widetilde{I}(c)$ for some cell c at any level $l < r+1$, then $i,j \in \widetilde{I}(c')$ for some cell c' at level $r+1$. Thus it suffices to consider the last level.

Fix any tape j for $1 \le j \le t$. Let C be an input dependency string of tape j at level $r+1$. From Proposition 3.2, C can be decomposed into a set S of t^{r+1} interleaved monotone sequences as described there. For any cell c on this tape and for any sequence $s \in S$, we say that c stands at some *stage* i in s if the rightmost entry before or in the cell portion of c in C that is assigned to s is i. Define a table $\mathcal{T}$ as follows. The rows of $\mathcal{T}$ correspond to the sequences in S and its columns correspond to the cells in tape j. Thus $\mathcal{T}$ has t^{r+1} rows and a number of columns equal to the

number of cells with nonempty input dependency, where the columns are also placed in left-to-right order as their corresponding cells. Each entry in $\mathcal{T}$ records the stage at which its corresponding cell stands in its corresponding sequence. For each column *col* corresponding to a cell c, we denote

$$H'(col) = \big\{\{\mathcal{T}_{row,col}\ ,\ \mathcal{T}_{row',col}\} \mid row \neq row' \text{ and } \mathcal{T}_{row,col} \neq \mathcal{T}_{row',col}\big\}.$$

Obviously $H(c) \subset H'(col)$ and $|H'(col)| \leq (t^{r+1})^2 = t^{2r+2}$.

For any two adjacent columns *col* and *col'* corresponding to two adjacent cells c and c', respectively, if $H'(col) \neq H'(col')$, then there must be some sequence $s \in S$ such that c and c' stand at different stages in s. Since s is monotone, this happens at most b times. Since there are at most t^{r+1} sequences, there are at most bt^{r+1} different $H'(col)$ over all columns. Thus

$$|\cup_{c\in\sigma(\mathbf{v})} H(c)| \leq t \cdot t^{2r+2} \cdot bt^{r+1} = bt^{3r+4}.$$

□

Proposition 3.4. *Suppose that an input* $\mathbf{v}$ *is partitioned into* $b = p \cdot m$ *blocks:* $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_{pm})$*, for some* $p \geq 2, m \geq 1$*. For any sequence of permutations* $\phi_1, \phi_2, \ldots, \phi_p$ *on* $[m]$*, there exists a set* $I \subset [m]$ *with* $|I| \geq m - p \cdot t^{5r+8}\mathsf{relsorted}(\{\phi_1, \ldots, \phi_p\})$ *satisfying the following property: for every* $i \in I$*, let* $J_i = \{\phi_1(i), m + \phi_2(i), \ldots, (p-1)m + \phi_p(i)\}$*, then there is no cell* c *in* $\sigma(\mathbf{v})$ *such that* $|J_i \cap \widetilde{I}(c)| > 1$*.*

Proof. This is a generalization of an argument in [GHS06] which gave a proof for the special case $p = 2$. As in the previous proposition, it suffices that we consider level $r+1$ only. Let $Q = \{i \in [m] \mid \exists c : |J_i \cap \widetilde{I}(c)| > 1\}$. Then $I = [m] \setminus Q$. Thus we need to prove $|Q| \leq p \cdot t^{5r+8}\mathsf{relsorted}(\{\phi_1, \ldots, \phi_p\})$.

We partition Q into disjoint subsets such that $Q = \cup_{1\leq p_1<p_2\leq p} Q_{p_1,p_2}$ and $Q_{p_1,p_2} \subseteq \{i \in Q \mid \exists c : (p_1-1)m + \phi_{p_1}(i), (p_2-1)m + \phi_{p_2}(i) \in \widetilde{I}(c)\}$ for any $1 \leq p_1 < p_2 \leq p$. By Proposition 3.3, there are at most pt^{3r+4} nonempty Q_{p_1,p_2}. Therefore there must be some $p_1 < p_2 \in [p]$ such that Q_{p_1,p_2} is of size at least $|Q|/\big(t^{3r+4}p\big)$. Fix these p_1 and p_2.

Let $Q_{p_1,p_2} = \{i_1, \ldots, i_q\}$, where $q = |Q_{p_1,p_2}|$. Let $C \in \{1, \ldots, pm\}^*$ obtained by concatenating the input dependency of all t tapes, so that for any cell c, if $\widetilde{I}(c)$ contains both $(p_1-1)m + \phi_{p_1}(i)$ and $(p_2-1)m + \phi_{p_2}(i)$ for some $i \in Q_{p_1,p_2}$, then both of them are placed consecutively and in this order. By Proposition 3.2, C can be decomposed into a set S of $t \cdot t^{r+1} = t^{r+2}$ monotone sequences. Let π be a permutation on $\{1, \ldots, q\}$ so that

$$(p_1-1)m + \phi_{p_1}(i_{\pi(1)}), (p_2-1)m + \phi_{p_2}(i_{\pi(1)}), \ldots, (p_1-1)m + \phi_{p_1}(i_{\pi(q)}), (p_2-1)m + \phi_{p_2}(i_{\pi(q)})$$

occur in the same order in C. Since there are q entries in input block p_1 and each of them must be in at least one sequence in S, there is some sequence $s \in S$ such that there are at least $q/|S| = q/t^{r+2}$ such entries in s. In other words, there exists a set $Q_1 \subset \{1, \ldots, q\}$ of size at least q/t^{r+2} such that for every $j \in Q_1$,

$$(p_1-1)m + \phi_{p_1}(i_{\pi(j)}) \in s,$$

and since s is monotone, for every $j_1 < j_2 \in Q_1$, either

$$\phi_{p_1}(i_{\pi(j_1)}) < \phi_{p_1}(i_{\pi(j_1)}) \text{ or } \phi_{p_1}(i_{\pi(j_1)}) > \phi_{p_1}(i_{\pi(j_1)}),$$

depending on the monotony of s. Let the indices in Q_1 be $j_1 < \ldots < j_{q_1}$. Consider the following list of entries

$$(p_2-1)m + \phi_{p_2}(i_{\pi(j_1)}), \ldots, (p_2-1)m + \phi_{p_2}(i_{\pi(j_{q_1})}),$$

which occurs in this order in C. As before, there must be at least one sequence $s' \in S$ such that there are at least $q_1/|S| = q_1/t^{r+2}$ such entries in s'. In other words, there exists a set $Q_2 \subset Q_1$ of size at least $q_1/t^{r+2} = q/t^{2r+4}$ such that for every $l \in Q_2$,

$$(p_2 - 1)m + \phi_{p_2}(i_{\pi(l)}) \in s',$$

and hence since s' is monotone, for every $l_1 < l_2 \in Q_2$, either

$$\phi_{p_2}(i_{\pi(l_1)}) < \phi_{p_2}(i_{\pi(l_1)}) \text{ or } \phi_{p_2}(i_{\pi(l_1)}) > \phi_{p_2}(i_{\pi(l_1)}),$$

depending on the monotony of s'. Therefore, by definition of sortedness,

$$\mathsf{relsorted}(\{\phi_1, \ldots, \phi_p\}) \geq \mathsf{sortedness}(\phi_{p_1}, \phi_{p_2}) \geq q/t^{2r+4},$$

which gives $q \leq \mathsf{relsorted}(\{\phi_1, \ldots, \phi_p\})t^{2r+4}$ and hence concludes the proposition. □

4 Simulation of read/write stream algorithms by communication protocols

Let $p \geq 2, m \geq 1$ and $\Phi = (\phi_1, \ldots, \phi_p)$ be a sequence of p permutations defined on $[m]$. Let X be a non-empty set and $Y = (Y_1, \ldots, Y_p) \in X^p$.

For each $i \in [m]$ and $\rho \in X^{(m-1)p}$, we define $J_i = \{\phi_1(i), m + \phi_2(i), \ldots, (p-1)m + \phi_p(i)\}$ and $\mathbf{v} = \mathbf{v}(Y, i, \rho, \Phi) = (\mathbf{v}_1, \ldots, \mathbf{v}_{pm}) \in X^{pm}$ such that $\mathbf{v}_{(j-1)m+\phi_j(i)} = Y_j$ for every $j \in [p]$, and $\mathbf{v}_{|\{1,\ldots,pm\}-J_i} = \rho$. Let A be a deterministic read/write stream algorithm defined on X^{pm}. Let $\sigma(\mathbf{v})$ be the dependency graph induced by A on $\mathbf{v}$ and $I_{\mathbf{v}} \subseteq [m]$ be the set of input indexes defined by $\sigma(\mathbf{v})$ as described in Proposition 3.4.

The following theorem will be used to show that by simulating an efficient deterministic read/write stream algorithm for $f^{\vee}_{\Phi}$, one can derive efficient p-party number-in-hand communication protocols for a variety of embeddings of f in $f^{\vee}_{\Phi}$.

Theorem 4.1. *Let $p \geq 2$, $m \geq 1$ and $\Phi = (\phi_1, \ldots, \phi_p)$ be a sequence of permutations on $[m]$. Let $X \neq \emptyset$ with $n = \lceil \log_2(X) \rceil$. Suppose that A is a deterministic (r, s, t)-algorithm defined on X^{pm}. Then there is a constant $c > 0$ depending on t such that, for each $i \in [m]$ and $\rho \in X^{p(m-1)}$ there is a p-party number-in-hand protocol $P_{i,\rho}$ in which player j has access to $Y_j \in X$ (and implicitly i and ρ) for each j, each $P_{i,\rho}$ communicates at most $O(t^{cr}p(s + \log pmn))$ bits, and for each i,ρ and $\mathbf{v} = \mathbf{v}(Y, i, \rho, \Phi)$ there is a set $I'_{i,\rho,\mathbf{v}} \subseteq [m]$ containing $I_{\mathbf{v}}$ such that if $i \notin I'_{i,\rho,\mathbf{v}}$ then $P_{i,\rho}$ on input $\mathbf{v}$ outputs "fail" and if $i \in I'_{i,\rho,\mathbf{v}}$ then $P_{i,\rho}$ on input $\mathbf{v}$ outputs the value $\mathsf{A}(\mathbf{v})$.*

Before going to the proof we need the following lemma.

Lemma 4.2. *When A terminates, the total length of all tapes used by A is at most $2^{(r+1)s}pmn$.*

Proof. The initial total length is clearly pmn. It is also clear that immediately after each reversal, the total length is multiplied by at most 2^s. The lemma follows. □

Proof of Theorem 4.1. We describe and then analyze the protocol. Each player first constructs $\mathbf{v} = \mathbf{v}(Y, i, \rho, \Phi)$ and then executes A on $\mathbf{v}$. Note that all players can access the whole input $\mathbf{v}$ except for the p blocks holding $Y_1, \ldots, Y_p$, each of which is known to exactly one player. Since no

player knows the whole input, in order to correctly simulate A, they need to communicate during the simulation. Along the way, each player gradually constructs and keeps a copy of $\sigma(\mathbf{v})$. Each keeps track of the level (the number of reversals) in $\sigma(\mathbf{v})$ that A is currently working on and the machine state of A. Essentially, for every tape cell at every level in $\sigma(\mathbf{v})$ written by A, the players record whether (1) the contents of the cell can be computed by everyone, or (2) the contents of the cell can only by computed by a specific player.

Those cells of type (1) are those cells c such that $(j-1)m+\phi_j(i) \notin \widetilde{I}_{pm}(c)$ for any $1 \leq j \leq p$. For each of these cells, each of the players records: the machine state immediately before overwriting the cell, and the (at most t) cells of the previous level on which this cell depends. Note that those cells that a cell of type (1) depends on are also type (1) cells. It is clear that by recursion, every player can compute the contents of each of these cells as needed.

Those cells of type (2) are those that depend on some input held by a particular player. Consider a cell c such that $(j-1)m + \phi_j(i) \in \widetilde{I}_{pm}(c)$ for some $j \in [p]$. Each player records that this cell depends on player j. We will show later what information player j needs to record so that she can compute the contents of c herself.

Note that there is another type of cell, whose contents depend on the inputs from more than one player. As soon as the simulation discovers one of these cells, it will stop and the protocol outputs "fail". We will explain more about this point later.

The simulation proceeds as follows. Each player executes A step by step. At every new step in which all the t tape heads are to read cells of type (1) only, every player can compute the contents of the t cells without any communication. Since each of them holds the current machine state, they can compute which one of the t tapes is written and the moves and the content of the write. Each of them thus records, for the overwritten cell, that it is of type (1) as well as the tape heads and the machine state. To end this step, each of the players also has the new machine state.

The more interesting case is when at a new step, at least one of the tape heads is to read at least one cell of type (2) and all of the type (2) cells depend on a player j. All players will then wait for player j to communicate. Player j will proceed as follows. As long as at least one of the tape heads still reads a cell depending on her or the algorithm does not make any reversal, she proceeds with the simulation, and clearly has sufficient information to do so. Along the way, for every cell she overwrites, she records the machine state and all the tape head positions for that cell, so that she can compute the cell later when needed. This process stops when the algorithm comes to a new step in which either all the tape heads are to read a cell of type (1), or at least one of the tape heads depends on another player, or one of the tape heads reverses its direction. When this process stops, player j broadcasts: (a) all t updated tape head positions and directions, and (b) the new machine state. Since there has been no reversal, all other players know precisely which cells were visited by player j and they mark all those overwritten cells, which are all of the same level in $\sigma(\mathbf{v})$, as of type (2) and depending on j. Therefore, all players now have sufficient information to proceed.

The last case is when at a new step, at least two of the tape heads are to read cells of type (2) and these two cells depend on two different players. In this case, all players stop the simulation and output "fail". It is clear that if $i \in I_{\mathbf{v}}$, by Proposition 3.4, this case will never happen.

When A terminates, the protocol will output exactly as A does. It remains to compute the communication cost.

We need to bound the cost of each communication and the number of times a communication occurs. From Lemma 4.2, the cost of one communication is $t\log(2^{rs}pn) + s = O(rs + \log pmn)$,

where the hidden constant depends on t.

When one player communicates, this means that one of the tape heads has either just moved out of a cell depending on her to another cell that is not, or has just reversed. The latter means the algorithm comes to the next level, which happens at most r times. By Proposition 3.1 (where we set $b = p$), the former occurs at most $t^{r+1} + t^r + \ldots + 1$ times for a single tape and single player. Summing over all tapes and all p players, this occurs at most pt^{r+3} times in total. □

5 Bounds for Disjointness and Frequency Moments

Using Lemma 2.3 and the fact that $\text{pDISJ}_{n,p}^{m,\Phi}$ is a special case of $\text{pDISJ}_{N,p}^{\Pi}$ where $N = mn$, we will lower bound the frequency moment problems by giving lower bounds for the $\text{pDISJ}_{n,p}^{m,\Phi}$ problem. This problem was previously studied in [BJR07] for the special case $p = 2$; our bounds extend those in [BJR07] for any $p \geq 2$ and also improve the bounds for $p = 2$.

Lemma 5.1. *There is a positive constant c such that given a randomized (r, s, t)-read/write stream algorithm A for $\text{pDISJ}_{n,p}^{m,\Phi}$ on X^{pm}, where $X = \{0,1\}^n$ and $\frac{pt^{5r+8}\mathsf{relsorted}(\Phi)}{m} = d < 1$, with error at most δ, there is a randomized public-coin p-party number-in-hand protocol P for $\text{pDISJ}_{n,p}$ of communication complexity $O(t^{cr}p(s + \log pmn))$ with error at most $\delta + d(1 - \delta)$.*

Proof. Suppose that A uses at most Γ random bits in its execution. For any string $\mathfrak{R} \in \{0,1\}^\Gamma$, let $\mathsf{A}_\mathfrak{R}$ denote the deterministic algorithm obtained from A using $\mathfrak{R}$ as source of randomness.

We consider the following public-coin randomized communication protocol $\mathcal{P}$. On inputs $Y = \{Y_1, \ldots, Y_p\} \in X^p$:

1. The players use the public coins to
   (a) uniformly and randomly generate $\rho \in f^{-1}(0)^{m-1} \subseteq X^{p(m-1)}$,
   (b) choose $1 \leq i \leq m$ uniformly and randomly,
   (c) uniformly and randomly generate a bit string $\mathfrak{R} \in \{0,1\}^\Gamma$,
   (d) uniformly and randomly generate a permutation $\phi : [n] \mapsto [n]$.
2. Each player j computes $Y'_j = \phi(Y_j)$.
3. The players run protocol $P_{i,\rho}$ based on $\mathsf{A}_\mathfrak{R}$ on inputs $Z = Z(Y'_1, \ldots, Y'_p, i, \rho, \Phi)$, as described in Theorem 4.1
4. If $P_{i,\rho}$ outputs "fail" then output 1; else output what $P_{i,\rho}$ does.

We analyze this protocol $\mathcal{P}$. Let $\sigma(\mathbf{v})$ be the dependency graph induced by $\mathsf{A}_\mathfrak{R}$ on input Z and $I_Z \subseteq [m]$ be the set of input indexes defined by $\sigma(\mathbf{v})$ as described in Proposition 3.4. Let $I' = I'_{i,\rho,Z}$ be the set of input positions induced by $\mathsf{A}_\mathfrak{R}$ given the choice of i, ρ and input Z as described in Theorem 4.1. The correctness of the protocol depends on whether $i \in I'$ and whether $\mathsf{A}_\mathfrak{R}(Z)$ is correct.

First we consider the case that the sets $Y_1, \ldots, Y_p$ are not disjoint, i.e $\text{pDISJ}_{n,p}(Y) = 1$. In this case if $P_{i,\rho}$ outputs "fail", the protocol always outputs correctly. Otherwise it will output what A does. In other words, we have

$$\begin{aligned}
\Pr[\mathcal{P}(Y)=1 \quad &| \quad \textsc{pDisj}_{n,p}(Y)=1] \\
&= \Pr[\mathcal{P}(Y)=1 \mid i\notin I', \textsc{pDisj}_{n,p}(Y)=1]\cdot\Pr[i\notin I' \mid \textsc{pDisj}_{n,p}(Y)=1] \\
&\quad + \Pr[\mathcal{P}(Y)=1 \mid i\in I', \textsc{pDisj}_{n,p}(Y)=1]\cdot\Pr[i\in I' \mid \textsc{pDisj}_{n,p}(Y)=1] \\
&= \Pr[i\notin I' \mid \textsc{pDisj}_{n,p}(Y)=1] + \Pr[\mathsf{A}_{\mathfrak{R}}(\mathbb{Z})=1 \text{ and } i\in I' \mid \textsc{pDisj}_{n,p}(Y)=1] \\
&\geq \Pr[\mathsf{A}_{\mathfrak{R}}(\mathbb{Z})=1 \mid \textsc{pDisj}_{n,p}(Y)=1] \geq 1-\delta.
\end{aligned}$$

Next we consider the case the sets $Y_1,\ldots,Y_p$ are disjoint. After being re-mapped by ϕ, the sets $Y'_1,\ldots,Y'_p$ are also disjoint and uniformly distributed in $f^{-1}(0)$. Hence Z is uniformly distributed over $(f^{-1}(0))^m$. Therefore I_Z is statistically independent from i. By Theorem 4.1, $|I'|\geq|I_Z|$ and by Proposition 3.4, $|I_Z|$ is always at least $(1-d)m$. Thus the probability that $i\in I'$ is at least $1-d$. Conditioned on this happening, the protocol outputs what A does, and hence errs with probability δ depending on $\mathfrak{R}$ only. In other words, we have

$$\begin{aligned}
\Pr[\mathcal{P}(Y)=0 \quad &| \quad \textsc{pDisj}_{n,p}(Y)=0] \\
&= \Pr[\mathcal{P}(Y)=0 \mid i\notin I', \textsc{pDisj}_{n,p}(Y)=0]\cdot\Pr[i\notin I' \mid \textsc{pDisj}_{n,p}(Y)=0] \\
&\quad + \Pr[\mathcal{P}(Y)=0 \mid i\in I', \textsc{pDisj}_{n,p}(Y)=0]\cdot\Pr[i\in I' \mid \textsc{pDisj}_{n,p}(Y)=0] \\
&= \Pr[\mathsf{A}_{\mathfrak{R}}(\mathbb{Z})=0 \mid i\in I', \textsc{pDisj}_{n,p}(Y)=0]\cdot\Pr[i\in I' \mid \textsc{pDisj}_{n,p}(Y)=0] \\
&\geq (1-\delta)\cdot(1-d).
\end{aligned}$$

This completes the lemma. □

Lemma 5.2. *Let $\delta<1/4$, $t\geq 1$, $1/2>\beta>0$, $\epsilon>\alpha>0$, and $m,n,p>1$ so that $p\leq m^{\frac{1}{2}-\alpha}=n^{\beta}$. Let $N=mn$. Then for large enough N, there is a constant $a>0$ with the following property: if $r\leq a\log N$ and $s=o(N^{1-\frac{4\beta}{2\beta+1}-\epsilon})$, then there is no randomized $\big(r,s,t\big)$-read/write stream algorithm with error at most δ for $\textsc{pDisj}_{n,p}^{m,\Phi^*}$ on $\{0,1\}^{pmn}$.*

Proof. Suppose by contradiction that there is such an algorithm A. By Lemma 5.1, there is a public-coin randomized number-in-hand communication protocol for $\textsc{pDisj}_{n,p}$ with complexity $O(t^{cr}ps)$ for some positive constant c with error at most $d+(1-d)\delta$, where $d=\frac{pt^{5r+8}\mathsf{relsorted}(\Phi^*)}{m}$. Since $r\leq a\log N=a(1+\frac{1-2\alpha}{2\beta})\log m$ and $\mathsf{relsorted}(\Phi^*)\leq 2e\sqrt{m}$, we have $d+(1-d)\delta\leq 2\delta<1/2$ for a sufficiently small depending on δ, t, and α.

By [CKS03], this communication complexity must be $\Omega(\frac{n}{p\log p})$. This gives us, for some constant ϵ'' depending on a and α that $s=\Omega(n^{1-2\beta-\epsilon''})=\Omega(N^{\frac{(1-2\beta-\epsilon'')(1-2\alpha)}{1+2\beta-2\alpha}})=\Omega(N^{1-\frac{4\beta}{2\beta+1}-\epsilon})$. □

This immediately implies a lower bound on $\textsc{pDisj}_{N,p}^{\Pi^*}$ where Π^* is the extension of Φ^* to a sequence of p permutations on $[N]$.

Theorem 5.3. *Let $\delta<1/4$, $t\geq 1$, $\epsilon>0$, and $N,p>1$ so that $p\leq N^{\gamma}$ for $\gamma<\frac{1}{4}$. Then for large enough N, there exists sequence Π^* of p permutations on $[N]$ such that there is no randomized $\big(o(\log N),N^{1-4\gamma-\epsilon},t\big)$-read/write stream algorithm with error at most δ for $\textsc{pDisj}_{N,p}^{\Pi^*}$.*

Proof. Follows from Lemma 5.2 where we set $N=mn$, $\Pi^*=(\pi_1^*,\ldots,\pi_p^*)$ to be the extension of Φ^* to p permutations on $[N]$ as described in Section 2, and set α,β so that $\frac{1}{\gamma}=\frac{1}{\beta}+\frac{2}{1-2\alpha}$ and α is sufficiently small. □

By Theorem 5.3 and Lemma 2.3(a) we obtain the following lower bounds for approximating F_k and F_∞^*.

Corollary 5.4. *Let $k > 1$, $t \geq 1$, $\eta > 0$, and $1 \geq \epsilon \geq 1/N$. Then there is no randomized $\big(o(\log N), O(\frac{1}{\epsilon^{4/k}} N^{1-\frac{4}{k}-\eta}), t\big)$-read/write stream algorithm that with probability at least $3/4$ outputs an approximation of F_k within a factor of $(1+\epsilon)$ on an input stream of up to N elements from $[N]$.*

Proof. Let ζ satisfy $\epsilon = N^{-\zeta}/4$. Then setting $p = \lceil N^{(1+\zeta)/k} \rceil = \lceil (4\epsilon N)^{1/k} \rceil$ and applying Lemma 2.3(a) and Theorem 5.3 we obtain that no randomized read/write stream algorithm with $o(\log N)$ reversals, t tapes, and space $O(N^{1-4(1-\zeta)/k-\eta})$ can compute F_k within a $(1+\epsilon)$ factor as described. Replacing $N^{-\zeta}$ by 4ϵ yields the claimed bound. □

Letting $\epsilon = 1$ in the above implies that there is no factor 2 approximation to F_k for $k > 4$ that uses small space and a small number of reversals in the read/write streams model.

Corollary 5.5. *Let $k > 4$, $t \geq 1$, and $\eta > 0$. Then there is no randomized $\big(o(\log N), O(N^{1-\frac{4}{k}-\eta}), t\big)$-read/write stream algorithm that with probability at least $3/4$ outputs an approximation of F_k within a factor of 2 on an input stream of up to N elements from $[N]$.*

By similar means we can derive improved lower bounds for computing F_∞^*.

Corollary 5.6. *Let $t \geq 1$, and $\eta > 0$. Then there is no randomized $\big(o(\log N), O(N^{1-\eta}), t\big)$-read/write stream algorithm that with probability at least $3/4$ outputs an approximation of F_∞^* within a factor of 2 on an input stream of up to N elements from $[N]$.*

We also derive lower bounds for the case that $k < 1$ using Theorem 5.3 and Lemma 2.3(b).

Corollary 5.7. *Let $k < 1$, $t \geq 1$, $\eta > 0$, and $1/N \leq \epsilon < 1/N^{3/4+\eta}$. Then there is no randomized $\big(o(\log N), O(\frac{1}{\epsilon^4 N^{3+\eta}}), t\big)$-read/write stream algorithm that with probability at least $3/4$ outputs an approximation of F_k within a factor of $(1+\epsilon)$ on an input stream of up to N elements from $[N]$.*

Proof. Define ζ so that $\epsilon = N^{-\zeta}/3$. Then for N sufficiently large as a function of k, if $p = N^{1-\zeta} = 3\epsilon N$ then $p - p^k \geq 2\epsilon N$ so Lemma 2.3(b) and Theorem 5.3 imply that no randomized read/write stream algorithm with $o(\log N)$ reversals, t tapes, and space $O(N^{1-4(1-\zeta)-\eta})$ can compute F_k within a $(1+\epsilon)$ factor as described. Replacing $N^{-\zeta}$ by 3ϵ yields the claimed bound. □

Our lower bound for $\textsc{pDisj}_{N,p}^{\Pi}$ is only interesting when $N = \omega(p^4)$.[2] This is because in order for the reduction from $\textsc{pDisj}_{n,p}^{m,\Phi}$ to work (Lemma 5.2), we need $N = nm$ and both $n = \omega(p^2)$ and $m = \omega(p^2)$. The condition $n = \omega(p^2)$ is induced by the communication complexity lower bound for $\textsc{pDisj}_{n,p}$, which is optimal up to a logarithmic factor. The following lemma shows that a condition requiring m to be polynomially larger than p is also necessary and that the above technique cannot yield bounds for constant factor approximations for $k < 3.5$.

[2]Note that if N is $O(p^2)$ there is a simple deterministic algorithm for $\textsc{pDisj}_{N,p}^{\Pi}$ for any Π. The algorithm first checks that the total size of the p subsets is at most N; otherwise it outputs 1. Then it scans for a subset of size s at most $N/p = O(p)$ and looks for these p elements in each of the other subsets in turn. It then splits these s elements into $p-1$ groups of consecutive elements of constant size and looks for the elements in the i-th group in the i-th of the other subsets. It is clear that the algorithm can be implemented using only two tapes, $O(\log n)$ space, and $O(1)$ reversals.

Lemma 5.8. *For integer $m < p^{3/2}/64$, any integer n, and for any $\Phi = (\phi_1, \ldots, \phi_p)$ defined on $[m]$, there is a deterministic $(2, O(\log(mnp)), 2)$-read/write stream algorithm computing* $\textsc{pDisj}_{n,p}^{m,\Phi}$.

To produce the algorithm claimed in Lemma 5.8, we need to show the following property of permutations that does not appear to have been considered previously. Its proof is inspired by Seidenberg's proof of the well-known theorem of Erdös and Szekeres (cf. [Ste95]) which shows that any pair of permutations must have relative sortedness at least $\sqrt{m}$. The difference is that with three permutations we can now ensure that the sequences appear in the same order in two of them rather than one possibly being reversed.

Lemma 5.9. *Let ϕ_1, ϕ_2, and ϕ_3 be permutations on $[m]$. Then there is some pair $1 \leq a < b \leq 3$ such that $\phi_a(1), \ldots, \phi_a(m)$ and $\phi_b(1), \ldots, \phi_b(m)$ have a common subsequence of length at least $m^{1/3}$.*

Proof. Suppose by contradiction that there are no such a and b. For every $i \in [m]$, let $\ell_i \in [m']^3$, where $m' = \lceil m^{1/3} - 1 \rceil$, be defined as follows: $\ell_i[1]$ is the length of the longest common subsequence of $\phi_1(1), \ldots, \phi_1(s)$ and $\phi_2(1), \ldots, \phi_2(t)$, where $\phi_1(s) = \phi_2(t) = i$, and $\ell_i[2]$ and $\ell_i[3]$ are defined analogously for the other two pairs ϕ_2, ϕ_3, and ϕ_1, ϕ_3, respectively.

Now for any $i \neq j \in [m]$, we must have $\ell_i \neq \ell_j$. This is because there must be some pair, say ϕ_1 and ϕ_2, such that either i occurs before j in both sequences or j occurs before i in both. In the first case $\ell_i[1] < \ell_j[1]$ and in the second case $\ell_i[1] > \ell_j[1]$.

However since $m' < m^{1/3}$, the number of different ℓ_i over all $i \in [m]$ is strictly less than m which is a contradiction. □

The above lemma is tight, even for any four permutations. An example of four permutations ϕ_1,ϕ_2, ϕ_3, and ϕ_4 of $[m]$ with the longest common subsequence between each pair being at most $m^{1/3}$ is as follows: ϕ_1 is the identity permutation, ϕ_2 is the increasing sequence of decreasing subsequences each of length $m^{2/3}$, ϕ_3 is the decreasing sequence of increasing sequences each of length $m^{1/3}$, and ϕ_4 is the decreasing sequence of increasing sequences of $m^{1/3}$ decreasing sequences each of length $m^{1/3}$. For example when $m = 8$, the sequences are $(1, 2, 3, 4, 5, 6, 7, 8), (4, 3, 2, 1, 8, 7, 6, 5), (7, 8, 5, 6, 3, 4, 1, 2)$, and $(6, 5, 8, 7, 2, 1, 4, 3)$.

Proof of Lemma 5.8. Given Φ there exist $J_1, J_2, \ldots, J_{p/3}$ defined as follows: J_1 is a common subsequence of two of ϕ_1, ϕ_2, and ϕ_3, indexed length at least $m^{1/3}$ given by Lemma 5.9; J_2 is a common subsequence of two of ϕ_4, ϕ_5, and ϕ_6 that is disjoint from J_1 and of length at least $(m - |J_1|)^{1/3}$; J_3 is a common subsequence of two of ϕ_7, ϕ_8, and ϕ_9 disjoint from $J_1 \cup J_2$ and of length at least $(m - |J_1| - |J_2|)^{1/3}$, and so on, with no index $i \in [m]$ appearing in more than one sequence. For each of the J_j let a_j and b_j denote the indices of the two permutations having the common subsequence J_j. The number of elements that do not appear in $J_1, \ldots, J_\ell$ is at most m_ℓ where m_ℓ is defined by the recurrence with $m_0 = m$ and $m_{j+1} = m_j - \lceil m_j^{1/3} \rceil$ for $j > 0$. If $m < p^{3/2}/64$, then $p \geq (64m)^{2/3} = 16m^{2/3}$. Now if $m_{p/8} > m/8$ then at least $(m/8)^{1/3} = m^{1/3}/2$ elements have been removed for each of $p/8 = 2m^{2/3}$ steps which implies $m_{p/8} = 0$, which is a contradiction. Repeating this argument reduces m_j to at most $m/64$ after another $2(m/8)^{2/3} = m^{2/3}/2 = p/32$ steps. Repeating this eventually yields that $m_{p/3} = 0$, which implies that every $i \in [m]$ is in exactly one J sequence.

The algorithm copies the input to tape 2 leaving both heads at the right end of the tape. It will use the head on tape 1 to scan the blocks for the players and the head on tape 2 to scan the corresponding blocks for the even-numbered players. It will solve the disjointness problems for

each block in the common subsequences $J_{p/3}, \ldots, J_2, J_1$, in turn. If an intersection is found in some pair of corresponding blocks in these sequences then the output is 1; otherwise, the output is 0. The promise ensures that, for each of the m subproblems, to check for a given common element it suffices to compare the blocks for a single pair of players. Since every $i \in [m]$ appears in some J_j, if we can position the heads to check the corresponding blocks then we can compute each of the m disjointness subproblems exactly and hence $\textsc{pDisj}^{m,\Phi}_{n,p}$.

It remains to show how the read/write stream algorithm positions the heads on the tapes in the positions corresponding to the sequences $J_{p/3}, \ldots, J_1$. These sequences can be hardwired into the state transition function as follows. We represent the sequence of positions indicated by $J_{p/3}, \ldots, J_1$ by a tuple of values that is stored internally by the algorithm as part of the state: $(j, i, a, k_a, b, k_b, f)$ where $j \in [p/3]$ denotes which J_j is being considered, $i \leq \lceil\sqrt{m}\rceil$ denotes the position within the common subsequence J_j that is being considered, a denotes the index of one of the two permutations that have J_j as a common subsequence, $k_a = k_a(j, i) \in [m]$ denotes the number of blocks in ϕ_a that must must be skipped over to get to the $i+1$-st position in the common subsequence, b denotes the index of the other of the two permutations that have J_j as a common subsequence, $k_b = k_b(j, i) \in [m]$ denotes the number of blocks ϕ_b that must must be skipped over to get to the $i + 1$-st position in the common subsequence, and f is a bit that will determine whether the algorithm is currently comparing positions and will be ready to have its heads repositioned when it has finished its comparison, or it is in the act of repositioning its heads. If $i + 1 \leq |J_j|$ the transition function will map $(j, i, a_j, 0, b_j, 0, 1)$ to $(j, i + 1, a_j, k_a(j, i + 1), b_j, k_b(j, i + 1), 0)$, leaving the remainder of the state unchanged; in this state it will move its heads to the left on tapes 1 and 2, decrementing the k_1 and k_2 counters, respectively, with each step. Otherwise the transition function will map it to $(j + 1, a_{j+1}, k_a(j + 1, 1), b_{j+1}, k_b(j + 1, i), 0)$ after skipping over the blocks for up to 2 ϕ_ℓ that are between ϕ_{a_j} and $\phi_{a_{j+1}}$ and between ϕ_{b_j} and $\phi_{b_{j+1}}$, respectively. After that it will move its heads to the left and decrement the counters as in the other case.

The total number of bits of state required is $O(\log pmn)$. □

6 Concluding remarks

We have introduced a novel reduction relating the read/write stream model directly to multiparty number-in-hand communication complexity. This reduction seems quite powerful and yields both simpler proofs and stronger results. It is likely to have useful implications for other problems.

An obvious open question is to close the gap between the upper and lower bounds for computing $\textsc{pDisj}^{\Pi}_{N,p}$ and approximating F_k. As we have shown, we are not far from the limit on lower bounds using the blocked and permuted version of disjointness, $\textsc{pDisj}^{m,\Phi}_{n,p}$; moreover, it is not clear how a simulation like that in Theorem 4.1 (which appears to be optimal up to a polylogarithmic factor) can be made to work for more general instances of $\textsc{pDisj}^{\Pi}_{N,p}$.

Optimizing our upper bound for $\textsc{pDisj}^{m,\Phi}_{n,p}$ raises the following interesting combinatorial question: Given a set of k permutations on $[m]$, what is the length of the longest common subsequence that can be guaranteed between some pair of these k permutations as a function of m and k? We conjecture that for any integer constants $m, c, k > 0$ such that $2^{c-1} < k \leq 2^c$, in any set of k permutations on $[m]$ there exists a pair such that the longest common subsequence of this pair of permutations has length $\Omega(m^{\frac{2^{c-1}-1}{2^c-1}})$. We can construct examples for any m, c, k to show that this bound cannot be improved, up to a constant factor. If this conjecture is correct, then our upper

bound for $\textsc{pDisj}_{n,p}^{m,\Phi}$ can be modified to match the lower bound, up to a $m^{o(1)}$ factor.

Acknowledgements

We would like to thank T.S. Jayram and Atri Rudra for helpful discussions.

References

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the 21st ACM Sympoisum on Principles of Database Systems (PODS)*, pages 1–16, 2002.

[BGKS06] L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. Simpler algorithms for estimating frequency moments of data streams. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 708–713, 2006.

[BJR07] P. Beame, T. S. Jayram, and A. Rudra. Lower bounds for randomized read/write stream algorithms. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 689–698, San Diego, CA, June 2007.

[BYJKS04] Z. Bar-Yossef, T.S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[CKS03] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.

[GHS06] Martin Grohe, Andre Hernich, and Nicole Schweikardt. Randomized computations on large data sets: Tight lower bounds. In *Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS)*, pages 243–252, 2006.

[GS05] Martin Grohe and Nicole Schweikardt. Lower bounds for sorting with few random accesses to external memory. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS)*, pages 238–249, 2005.

[IW03] Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 283–292, 2003.

[IW05] Piotr Indyk and David P. Woodruff. Optimal approximations of frequency moments of data streams. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC)*, pages 202–208, 2005.

[Mut06] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2006.

[Raz92] A. A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

[SS02] M. E. Saks and X. Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, Montreal, Quebec, Canada, May 2002.

[Ste95] J. M. Steele. Variations on the monotone subsequence problem of Erdös and Szekeres. In Aldous, Diaconis, and Steele, editors, *Discrete Probability and Algorithms*, pages 111–132. Springer-Verlag, 1995.

[Ste97] J. M. Steele. *Probability Theory and Combinatorial Optimization.* SIAM, Philadelphia, 1997.

[Woo04] David P. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 167–175, 2004.

ECCC ISSN 1433-8092

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