The Value of Multiple Read/Write Streams for Approximating Frequency Moments

PAUL BEAME and TRINH HUYNH
University of Washington

We consider the read/write streams model, an extension of the standard data stream model 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 multiple 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 sub-logarithmic 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 one-pass data stream there is a randomized constant-factor approximation for F_k using $\tilde{O}(N^{1-2/k})$ space, and that by sorting, which can be done deterministically in $O(\log N)$ passes using 3 read/write streams, F_k can be computed exactly. 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.

We obtain our results by showing a new connection between the read/write streams model and the multiparty number-in-hand communication model.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Bounded-action devices*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Relations between models*; E.4 [**Coding and Information Theory**]: *Formal models of communication*; H.2.4 [**Database Management**]: Systems—*Query processing*

General Terms: Theory, Algorithms

Additional Key Words and Phrases: data streams, read/write streams, external-memory algorithms, frequency moments, communication complexity

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.

A preliminary version of this paper, [Beame and Huynh-Ngoc 2008], appeared in the Proceedings of the *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*.
Authors' address: Computer Science and Engineering, University of Washington, Seattle, WA 98195-2350.
The research of the first author was supported by NSF grants CCF-0514870 and CCF-0830626. The research of the second author was supported by a Vietnam Education Foundation Fellowship and by NSF grant CCF-0830626.
The second author is also known as Dang-Trinh Huynh-Ngoc.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

However, as Grohe and Schweikardt [2005] observed, in many natural situations for which the data stream model has been studied, the computation also has access to very large 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 [Beame et al. 2007], 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 [Grohe and Schweikardt 2005], 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. 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 [Grohe and Schweikardt 2005; Grohe et al. 2006; 2009] show that such small space read/write stream algorithms are not possible using fewer passes; moreover, lower bounds for the related problems of determining whether two sets are equal, or intersect, or of determinining whether or not the input stream consists of distinct elements have also been shown [Grohe et al. 2006; Beame et al. 2007; Grohe et al. 2009].

While these lower bounds give us significant understanding of the read/write streams model, they apply to exact computation and do not say much about the potential of the read/write streams model for more efficient solutions of approximation problems, which are the major successes of the standard data stream model (see surveys [Babcock et al. 2002; Muthukrishnan 2006]). Among the most notable successes are the surprising one-pass small space randomized algorithms for approximating the *frequency moments* of data streams first shown by Alon et al. [1999]. 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. As special cases, these frequency moments include the length of the data stream (F_1), the number of distinct elements in the stream (F_0), the size of the self-join on the keys of a database relation represented by the stream (F_2), or the largest frequency of any element in the stream (F^*_∞). Alon et al. [1999] gave algorithms to approximate frequency moments F_k within a constant $(1+\epsilon)$ factor on streams of length N using $\tilde{O}(N^{1-1/k})$ space[1] for $k > 2$ and $O(\log N)$ space for $k \leq 2$ [Alon et al. 1999]. Subsequently, the resources required for $(1+\epsilon)$ factor approximations of F_k for $k > 2$ have been improved to $\tilde{O}(N^{1-2/k})$ space [Indyk and Woodruff 2005; Bhuvanagiri et al. 2006].

Alon et al. [1999] also showed that their algorithms were not far from optimal in the data stream model; in particular, by extending bounds in [Razborov 1992] for the randomized 2-party communication complexity of a promise version of the set disjointness problem from 2 to p players where each player has access to its own private portion of the input (a model known as the *p-party number-in-hand communication game*), they showed that F_k requires $\Omega(N^{1-5/k})$ space to approximate by randomized algorithms. A series of papers [Saks and Sun 2002; Bar-Yossef et al. 2004; Chakrabarti et al. 2003; Gronemeier 2009; Jayram 2009] has improved the space lower bound for fixed ϵ to $\Omega(N^{1-2/k})$, nearly matching the upper bound, by finding an optimal lower bound for the communication complexity of set disjointness. Thus, F_k for $k > 2$ requires polynomial space in the data stream model[2].

This leads to the natural questions: Can one prove good lower bounds for approximation problems in the read/write streams model? Can read/write stream algorithms approximate larger frequency moments more efficiently than one-pass algorithms can?

[1]The notation $\tilde{O}(\cdot)$ here suppresses logarithmic factors in N and polynomial factors in $1/\epsilon$.

[2]There is also a $\Theta(1/\epsilon^2)$ dependence of the space on the error in the approximation. This was shown to be optimal for one-pass algorithms [Indyk and Woodruff 2003; Woodruff 2004] and multiple-pass data stream algorithms [Chakrabarty and Regev 2011], though by considering a different two-party communication problem – gap Hamming distance.

We show that the ability to augment the data stream model with multiple read/write streams does not produce significant additional efficiency in approximating frequency 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 constant $\delta > 0$. This lower bound is very similar to the upper bound even for ordinary one-pass read-only data streams (and is larger than the original lower bound in [Alon et al. 1999] for such ordinary data streams).

The major difficulty in developing lower bounds for the read/write streams model, in contrast to the data stream model, is that an easy reduction from number-in-hand multiparty communication complexity breaks down. This fails for read/write stream algorithms because different parts of the computations can communicate with each other by writing to the streams. In fact, as we observe in Section 2, the (unique-intersection) p-party promise set-disjointness problem, which is the basis for the lower bounds for approximating frequency moments in the data stream model, can be easily solved by a read/write stream algorithm 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 been based on special-purpose combinatorial methods developed especially for the model.

Grohe, Hernich, and Schweikardt [Grohe and Schweikardt 2005; Grohe et al. 2006; 2009] identified certain structural properties of the executions of read/write stream algorithms, their *skeletons*, and applied cut-and-paste arguments along with these skeletons to show the existence of certain combinatorial rectangles on which the algorithms' answers must be constant. They showed that the existence of these rectangles implies that no space-efficient read/write stream algorithm can sort in $o(\log N)$ passes or determine, with one-sided error bounded below 1/2, whether or not two input sets are equal. Then, by reduction, they derived lower bounds for one-sided error randomized algorithms for several other problems.

Beame et al. [2007] used the structure of the rectangles produced in [Grohe and Schweikardt 2005; Grohe et al. 2006; 2009] together with additional combinatorial reasoning to show how standard properties that lower bound randomized two-party communication complexity – discrepancy and corruption over rectangles – can be used to derive lower bounds for randomized read/write streams with two-sided error. Using this approach they gave 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 several 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 [Grohe and Schweikardt 2005; Grohe et al. 2006; 2009; Beame et al. 2007] do not yield lower bounds on the approximate computation of frequency moments F_k for any $k < \infty$. In particular it is consistent with all previous work that read/write stream algorithms can compute constant factor approximations to any such F_k using $o(\log N)$ passes, $O(\log N)$ space, and only 2 streams. We show that this is not possible.

We take a different approach to lower bounds in the read/write streams model from the approaches in previous work. Despite the failure of the standard reduction, we are able to characterize read/write stream algorithms via a 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 [Grohe and Schweikardt 2005; Grohe et al. 2006; 2009]. 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 [Beame et al. 2007] but extended to the case of p sets. This allows us to obtain our $\Omega(N^{1-4/k-\delta})$ 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 a gap between our read/write streams lower bounds and the data stream upper bounds; our lower bounds are limited by the relationship between the number of blocks and the number of sets in the permuted disjointness problem that we consider. We also show that modifying this relationship cannot improve the lower bound for constant factor approximations for $k < 3.5$. In particular, there is a deterministic read/write stream algorithm with three passes, two streams and $O(\log N)$ space that can compute the value of the blocked and permuted p-party disjointness problem for any numbers of blocks and sets that would have produced such lower bounds. To derive this algorithm we show a novel property of the lengths of common subsequences in sets of permutations.

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 the read/write streams algorithm can be randomly accessed.

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, the remaining tapes blank, and each read/write head at the start of its tape. 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 $\mathbf{v} \in \Sigma^N$, the total number of reversals of the direction of movement of all heads is at most r.

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 $[m]$ and for any permutation ϕ of $[m]$, define the *sortedness* of ϕ, denoted by $\mathsf{sortedness}(\phi)$, to be the length of the longest monotone subsequence of $(\phi(1), \ldots, \phi(m))$. Thus, in particular, $\mathsf{sortedness}(\phi_1\phi_2^{-1})$ is the length of the longest common subsequence of $(\phi_1(1), \ldots, \phi_1(m))$ and $(\phi_2(1), \phi_2(2), \ldots, \phi_2(m))$, or of $(\phi_1(1), \ldots, \phi_1(m))$ and $(\phi_2(m), \phi_2(m-1), \ldots, \phi_2(1))$, whichever is greater. For convenience of notation, we write $\mathsf{sortedness}(\phi_1, \phi_2)$ for $\mathsf{sortedness}(\phi_1\phi_2^{-1})$. For any $m, p \geq 1$ and a sequence $\Phi = (\phi_1, \phi_2, \ldots, \phi_p)$ of per-

mutations of $[m]$, define the *relative sortedness* of Φ, denoted by $\mathsf{relsorted}(\Phi)$, to be $\max_{i \neq j \in [p]}(\mathsf{sortedness}(\phi_i, \phi_j))$.

LEMMA 2.1 (CF. [STEELE 1997]; 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 $p \leq m^c$ for some constant $c > 0$ and m is sufficiently large, there exists a sequence $\Phi^*_{p,m} = (\phi_1, \phi_2, \ldots, \phi_p)$ of permutations of $[m]$ such that* $\mathsf{relsorted}(\Phi) < 2e\sqrt{m}$.

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 $2 \leq p < n$, define the promise problem $\textsc{pDisj}_{n,p} : \{0,1\}^{np} \to \{0,1\}$ as follows: For $x_1, \ldots, x_p \in \{0,1\}^n$, 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, $\textsc{pDisj}$ is undefined.

We use the usual definition of p-party number-in-hand communication complexity. A series of communication complexity lower bounds [Alon et al. 1999; Saks and Sun 2002; Bar-Yossef et al. 2004; Chakrabarti et al. 2003; Gronemeier 2009; Jayram 2009] for $\textsc{pDisj}_{n,p}$ in this number-in-hand model has resulted in the essentially optimal lower bounds for computing frequency moments in the data stream model, even allowing multiple passes over the input stream. Gronemeier [2009] and Jayram [2009] gave the strongest of these results showing that any p-party public-coin randomized number-in-hand communication protocol for $\textsc{pDisj}_{n,p}$ must have complexity at least $\Omega(n/p)$, which is optimal.

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 $\textsc{pDisj}_{n,p}$: Copy x_1 to the second tape and compare the contents of x_1 and x_2 bit-by-bit using the two tape heads. The promise nature of the problem ensures that the output is completely determined by any pair of x_i and x_j, where $i \neq j$. 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 $\textsc{pDisj}^\Pi_{N,p} : \{0,1\}^{Np} \to \{0,1\}$ 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 in [Alon et al. 1999] yields lower bounds for F_k given lower bounds for $\textsc{pDisj}^\Pi_{N,p}$ for suitable choices of p. We note that the following lemma applies similarly to F^*_∞, where $1/\infty$ is interpreted as 0.

LEMMA 2.3. *Let $p, N \geq 2$, $1/2 > \epsilon, \delta \geq 0$, and $k \geq 0$ satisfy*

(a) if $k > 1$, then $p \geq c(\epsilon N)^{1/k}$ for $c > 0$ and $c^k > c + 3$, and
(b) if $k < 1$, then $p \geq c\epsilon N$ for $c > 0$ and $c > c^k + 3$.

Suppose that there is a randomized (r, s, t)-read/write stream algorithm A that outputs an $(1+\epsilon)$-approximation of F_k on every sequence of at most N elements in $[N]$ with probability at least $1 - \delta$. Then for any $\Pi = (\pi_1, \ldots, \pi_k)$ sequence of permutations of $[N]$, there is a randomized $(r, s + \log_2 N, t)$-read/write stream algorithm A' that solves $\textsc{pDisj}^\Pi_{p,N}$ with error at most δ.

PROOF. The algorithm A' is obtained from A by implicitly converting every $y_{i,j}$ input bit for $\textsc{pDisj}^\Pi_{p,N}$ that is 1 into an input number $\pi_i(j) \in [N]$ for the F_k problem (here, we

can assume that A' can compute $\pi_i(j)$ for free since Π is fixed). With $\log_2 N$ extra bits of memory, A' can also check if the total number N' of 1-bits in all p vectors is larger than N. If $N' > N$, then because of the promise of $\text{pDISJ}_{p,N}$, the sets must be intersecting, and hence A' outputs 1 and halts. We now assume that $N' \leq N$. Note that by the promise, when $\text{pDISJ}_{p,N} = 0$, we have $F_k = N'$.

Next we consider part (a). When $\text{pDISJ}_{p,N} = 1$, we have that by the definition of F_k,

$$\begin{aligned}
F_k &= N' + p^k - p \\
&\geq N' + c^k \epsilon N - c(\epsilon N)^{1/k}, \qquad \text{since } p \geq c(\epsilon N)^{1/k} \text{ and } k > 1, \\
&\geq N' + 3\epsilon N' + (c^k - 3)\epsilon N - c(\epsilon N)^{1/k}, \qquad \text{since } N \geq N', \\
&\geq (1+\epsilon)^2 N' + (c^k - 3)\epsilon N - c(\epsilon N)^{1/k}, \qquad \text{since } 0 \leq \epsilon < 1/2, \\
&> (1+\epsilon)^2 N', \qquad \text{since } c^k > c + 3.
\end{aligned}$$

Then A' just outputs 1 if F_k is greater than $(1+\epsilon)N'$ and outputs 0 otherwise. The correctness follows from that of A.

Next we consider part (b). We proceed as in (a), except that now the premises yield different implications for the relative values of F_k and therefore the output condition will be different. If $\text{pDISJ}_{p,N} = 0$ then $F_k = N'$ as before. Since $k < 1$, if $\text{pDISJ}_{p,N} = 1$ then now $F_k < N'$. In particular,

$$\begin{aligned}
F_k &= N' - p + p^k \\
&\leq N' - c\epsilon N + (c\epsilon N)^k, \qquad \text{since } p \geq c\epsilon N \text{ and } k < 1, \\
&\leq N' - 3\epsilon N' - (c-3)\epsilon N + (c\epsilon N)^k, \qquad \text{since } N \geq N', \\
&\leq N'/(1+\epsilon)^2 - (c-3)\epsilon N + (c\epsilon N)^k \qquad \text{since } 0 \leq \epsilon < 1/2, \\
&< N'/(1+\epsilon)^2 \qquad \text{since } c - 3 > c^k.
\end{aligned}$$

Then A' outputs 1 if the value returned for F_k is smaller than $N'/(1+\epsilon)$ and output 0 otherwise. □

For our lower bound arguments we will need the sequence of permutations Π to be 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 $j' \in [m]$ and $j'' \in [n]$. 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 $\text{pDISJ}^{m,\Phi}_{n,p}$ 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 [Beame et al. 2007], we see that $\text{pDISJ}^{m,\Phi}_{n,p} = f^g_\Phi$ where $f = \text{pDISJ}_{n,p}$ and $g = \vee$, and

$$f^g_\Phi(\mathbf{v}_{11}, \ldots, \mathbf{v}_{1m}, \ldots, \mathbf{v}_{p1}, \ldots, \mathbf{v}_{pm}) = g\big(\ldots, f(\mathbf{v}_{1\phi_1(j)}, \ldots, \mathbf{v}_{p\phi_p(j)}), \ldots\big),$$

where j runs from 1 to m.

3. DEPENDENCY GRAPHS AND INFORMATION FLOW IN READ/WRITE STREAM ALGORITHMS

Our main goal is to argue that functions of the form f^g_Φ, such as $\text{pDISJ}^{m,\Phi}_{n,p}$, can be hard to solve in the read/write streams model, even though the ordinary un-permuted version f^g, which corresponds to $\text{pDISJ}_{n,p}$, may be easy. The intuition is that since the input bits are permuted, an algorithm with small space has to make many passes, back and forth, to check

corresponding bits in different input vectors. To this end, in this section we define and study an object called a *dependency graph* to capture the flow of information between the tapes in various stages of a deterministic read/write stream algorithm's execution. This notion simplifies the notion of skeletons used in previous work [Grohe and Schweikardt 2005; Grohe et al. 2006; 2009; Beame et al. 2007]; although much simpler, it suffices for our purposes. In this section, we only consider deterministic (r, s, t)-read/write stream algorithms.

Recall that the input is initially on the first tape, and that at any step, one of the t tape heads overwrites the contents of the current cell and moves either left or right, all of which depend only on the state of the algorithm and the contents of the t cells currently read by the heads.

Definition 3.1. Fix a deterministic read/write stream algorithm A that makes r reversals on input $\mathbf{v}$. The dependency graph corresponding to $\mathbf{v}$, denoted by $\mathcal{G}(\mathbf{v})$, has $r+2$ levels; level 0 corresponds to the beginning of the computation and level $r+1$ corresponds the end of the computation. Level ℓ for $1 \leq \ell \leq r$ encodes the dependency on the input of each of the t tapes immediately before the ℓ-th reversal in the following manner: For $0 \leq \ell \leq r+1$ there is one node at level ℓ of $\mathcal{G}(\mathbf{v})$ 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 ℓ-th reversal, or before the end of the computation for $\ell = r+1$. The nodes at level ℓ are ordered according to the positions of their corresponding cells on the tapes. Because of this we can view nodes of the dependency graph, interchangeably, as tape cells. There are directed edges $u \to v$ to each node v at level ℓ from each node u at level $\ell-1$ that v *depends* on, as described below. (We will drop $\mathbf{v}$ from $\mathcal{G}(\mathbf{v})$ if the input is fixed.)

The crucial observation made in [Grohe and Schweikardt 2005] 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 $(\ell-1)$-st and ℓ-th reversal (i.e, at level ℓ of $\mathcal{G}(\mathbf{v})$), 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 $(\ell-1)$-st reversal. This implies that any cell at level ℓ depends either on t cells including itself in level $\ell-1$ (when it is overwritten in level ℓ) or only on itself in level $\ell-1$ (when it is intact in level ℓ). 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.

Next we define the notion of *input dependency*. The definition is motivated by the fact that the input to a function f_Φ^g is partitioned into blocks.

Definition 3.2. For a fixed input $\mathbf{v}$ and some $b \geq 1$, let $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_b)$ be the partition of $\mathbf{v}$ into b blocks of consecutive bits. For every cell/node c in $\mathcal{G}(\mathbf{v})$, define the *input dependency* of c, denoted $\mathcal{I}_b(c) \subseteq [b]$, to be the set of input blocks that it *depends* on: $i \in \mathcal{I}_b(c)$ if and only if there is a directed path from a cell in $\mathbf{v}_i$ at level 0 to c.

We note that the set $\mathcal{I}_b(c)$ 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. Since the fan-in of every node in the graph is at most t, we immediately have that for every cell c in level ℓ, $|\mathcal{I}(c)| \leq t^\ell$.

We will prove two key lemmas in this section, Lemma 3.4 and Lemma 3.8. The first lemma, Lemma 3.4, says that for most nodes/cells in $\mathcal{G}(\mathbf{v})$, if the node/cell depends on some input block, then so are adjacent cells. Before proving this lemma, we make the following basic observation. For any cell c, we write $right(c)$ and $left(c)$ for the cells immediately to its right and to its left, respectively, on the same tape.

PROPOSITION 3.3. *Consider any fixed level $0 < \ell \leq r+1$ in $\mathcal{G}$. Fix any cell c on the j-th tape at level ℓ, and cell $\tilde{c}$ on the $\tilde{j}$-th tape at level $\ell-1$, for some $j, \tilde{j} \in [t]$, such that $\tilde{c} \to c$ is in $\mathcal{G}$. Assume that both cells c and $right(c)$ are overwritten in the ℓ-th pass, then the following holds:*

- *if the j-th and $\tilde{j}$-th heads are moving in the same direction in the ℓ-th pass, then there is a unique cell c' on the $\tilde{j}$-th tape from $\tilde{c}$ to the right such that $c' \to right(c)$ is in $\mathcal{G}$,*
- *and if the j-th and $\tilde{j}$-th heads are moving in opposite direction in the ℓ-th pass, then there is a unique cell c' on the $\tilde{j}$-th tape from $\tilde{c}$ to the left such that $c' \to right(c)$ is in $\mathcal{G}$.*

Similarly by symmetry, the above statements are true if we simultaneously replace every "right" with "left" and every "left" with "right".

PROOF. We prove the first item, with the second item followed by symmetry. For the first item, there are 2 cases: (I) either both heads are moving from left to right, or (II) they are moving from right to left in the ℓ-th pass.

Let us first consider case (I). Since $\tilde{c} \to c$ is in $\mathcal{G}$, when the cell c is overwritten, the head on the $\tilde{j}$-th tape is reading $\tilde{c}$. After c is overwritten, the j-th tape head moves to $right(c)$. Since we assume that $right(c)$ will be overwritten, the claim follows.

The claim for case (II) also follows by a similar observation. □

LEMMA 3.4. *Suppose that an input $\mathbf{v}$ is partitioned into b blocks: $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_b)$. Let C be the set of all cells on the j-th tape at any level ℓ in $\mathcal{G}(\mathbf{v})$, for $0 \leq \ell \leq r+1$ and any $j \in [t]$. Fixing any $i \in [b]$, let $S_{j,\ell} = \{c \in C \mid i \in \mathcal{I}(c) \text{ and } i \notin \mathcal{I}(right(c))\}$ and $S'_{j,\ell} = \{c \in C \mid i \in \mathcal{I}(c) \text{ and } i \notin \mathcal{I}(left(c))\}$. Then $|S_{j,\ell}|$ and $|S'_{j,\ell}|$ are both $\leq t^{\ell+1}$.*

PROOF. We will bound the size for $S_{j,\ell}$, and the bound for $S'_{j,\ell}$ will follow by symmetry.

The lemma holds for $\ell = 0$ since the only cell at level 0 in $S_{j,0}$ is at the right boundary of $\mathbf{v}_i$. Thus $|S_{j,0}| = 1$. Furthermore, if there is only 1 tape ($j = t = 1$), then it is immediate that this is also the only cell at any level $\ell > 0$ in $S_{1,\ell}$. Thus $|S_{1,\ell}| = 1$ for any $\ell \geq 0$ if $t = 1$. Thus in the following, it suffices to prove the lemma for $t > 1$ and $\ell > 0$. We will show that $|S_{j,\ell}| \leq t^\ell + t^{\ell-1} + \ldots + 1$ and thus is $< t^{\ell+1}$.

Proceding by induction on ℓ, consider any fixed $\ell > 0$. Recall that at level ℓ, a cell in $\mathcal{G}(\mathbf{v})$ gets its input dependency from at most t cells on different tapes at level $\ell - 1$ that it depends on. To bound $|S_{j,\ell}|$, we partition $S_{j,\ell} = U \cup V \cup W$, where U consists of cells $c \in S_{j,\ell}$ that are not overwritten in the ℓ-th pass, V consists of cells $c \in S_{j,\ell}$ that are overwritten but $right(c)$ is not overwritten in the ℓ-th pass, and W consists of cells $c \in S_{j,\ell}$ such that both c and $right(c)$ are overwritten in the ℓ-th pass.

We observe that $|V| \leq 1$ since the only cell in V is the rightmost cell on tape j that is overwritten in the ℓ-th pass. We also observe that any $c \in U$ also appears in $S_{j,\ell-1}$ (at level $\ell - 1$). This follows since if $c \in U$, then the same cell c at level $\ell - 1$ also depends on the input block i, whereas the cell $right(c)$ at level $\ell - 1$ does not depend on this input block (since $right(c)$ at level ℓ does not). Thus $c \in S_{j,\ell-1}$.

Next we bound $|W|$. Consider any $c \in W$. Since c is overwritten, there must be some cell $\tilde{c}$ from level $\ell-1$ on an $\tilde{j}$-th tape such that $\tilde{c} \to c$ is in $\mathcal{G}$, and $i \in \mathcal{I}(\tilde{c})$. By Proposition 3.3, since $right(c)$ is also overwritten, there is a unique cell c' on the $\tilde{j}$-th tape such that $c' \to right(c)$ is in $\mathcal{G}$. Furthermore, c' must be from $\tilde{c}$ to the right or to the left, depending only on the relative movements of the j-th and $\tilde{j}$-th heads. Without loss of generality, assume the two heads move in the same direction. Since $i \notin \mathcal{I}(right(c))$, we have $i \notin \mathcal{I}(c')$. Thus, there must be a cell c'' between $\tilde{c}$ and c' (also on level $\ell - 1$) such that $i \in \mathcal{I}(c'')$ but $i \notin \mathcal{I}(right(c''))$. In other words, $c'' \in S_{\tilde{j},\ell-1}$. We say that c'' *contributes* to $c \in W$.

Note that in any fixed level $\ell > 0$, no head reverses its direction. Thus, any such cell c'' from level $\ell - 1$ contributes to at most one cell $c \in W$. Summing over all t tapes at level $\ell - 1$, we have that $|W| \leq |S_{1,\ell-1}| + \ldots + |S_{t,\ell-1}|$.

Thus $|U \cup V \cup W| \leq |S_{j,\ell-1}| + 1 + |S_{1,\ell-1}| + \ldots + |S_{t,\ell-1}|$. However, with only one head on each tape, a cell c'' on the same tape $\tilde{j} = j$ can only contribute to the same cell $c = c'' \in W$, and thus, any cell in $S_{j,\ell-1}$ (at level $\ell - 1$) which is also in U (at level ℓ) cannot contribute to any cell in W. Thus we have

$$\begin{aligned} |S_{j,\ell}| &= |U \cup V \cup W| \\ &\leq 1 + |S_{1,\ell-1}| + \ldots + |S_{t,\ell-1}|, \end{aligned}$$

then by induction,

$$\begin{aligned} &\leq 1 + t(t^{\ell-1} + t^{\ell-2} + \ldots + 1) \\ &\leq t^{\ell} + t^{\ell-1} + \ldots + 1, \end{aligned}$$

which completes the proof. □

Before going to the second key lemma, Lemma 3.8, we define some more notation and make a few more observations. 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 $j \in Q_i$, we say that the string s_i *takes* the j-th entry of s.

Consider any fixed dependency graph $\mathcal{G}(\mathbf{v})$ on an input $\mathbf{v} = \mathbf{v}_1 \cdots \mathbf{v}_b$. For any fixed level ℓ in $\mathcal{G}(\mathbf{v})$ and any fixed tape, consider the sequence of nodes in $\mathcal{G}(\mathbf{v})$ of all the cells on the tape, in left to right order. An *input dependency string* C of this tape at this level is any string in $\mathcal{S}_{\mathcal{I}(c_1)} \cdots \mathcal{S}_{\mathcal{I}(c_L)} \subset [b]^*$ where the cells of the tape are $c_1, \ldots, c_L$, in left to right order. Thus C can be partitioned into L disjoint substrings, each of which corresponds to a string in $\mathcal{S}_{\mathcal{I}(c_i)}$ for some $i \in [L]$, and is called a *cell portion* of C associated with the cell c_i.

PROPOSITION 3.5. *Let $C \in [b]^*$ be an input dependency string of any one tape at any level ℓ in $\mathcal{G}(\mathbf{v}_1 \cdots \mathbf{v}_b)$. Then C can be written as the interleaving of at most t^{ℓ} monotone sequences so that every such sequence s takes at most one entry in every cell portion of C.*

PROOF. The general idea is taken from [Grohe et al. 2006; 2009]. We proceed by induction on ℓ. We will prove a somewhat stronger inductive claim, namely that the property above also holds for more general strings, namely those strings C in $\bigcup_{a_i \geq 1} \mathcal{S}^{a_1}_{\mathcal{I}(c_1)} \cdots \mathcal{S}^{a_L}_{\mathcal{I}(c_L)}$, 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 $\mathcal{S}_{\mathcal{I}(c_i)}$ for some $i \in [L]$.

For $\ell = 0$, the only non-empty tape is the input tape and C itself is a monotone sequence $1 \cdots 1 2 \cdots 2 \cdots b \cdots b$ of $|\mathbf{v}|$ entries (where "1" occurs $|\mathbf{v}_1|$ times, "2" occurs $|\mathbf{v}_2|$ times, and so on). Obviously, every nonempty cell c has $|\mathcal{I}(c)| = 1$. Thus the claim is true for $\ell = 0$.

For the induction step, suppose that the tape we are considering is the j-th tape, where $j \in [t]$. At level ℓ, the j-th tape head visits some consecutive cells in the j-th 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 $\ell - 1$ and D corresponds to those cells that are visited. For each of those former cells, its input dependency is unchanged from level $\ell - 1$, and for each of those latter cells, its input dependency is the union of those of the t cells from level $\ell - 1$ that it depends on. Thus D can be written as the interleaving of t sequences $D_1, \ldots, D_t$, where sequence D_i for $i \in [t]$ denotes a substring of an extended input dependency of tape i from level $\ell - 1$, or the reverse of it, depending only on whether the j-th and the i-th heads are moving in the same or opposite direction.

Note that $C'D_jC''$ is an extended input dependency string of tape j at level $\ell-1$. By induction, each of $D_1,\ldots,D_t$ and $C'D_jC''$ can be written as the interleaving of at most $t^{\ell-1}$ monotone sequences each of which takes at most 1 entry from every cell portion. Hence C can be written as the interleaving of at most t^ℓ monotone sequences satisfying the same requirement. □

PROPOSITION 3.6. *Suppose that an input* $\mathbf{v}$ *is partitioned into* b *blocks* $\mathbf{v}=(\mathbf{v}_1,\ldots,\mathbf{v}_b)$ *for some* $b\geq 1$. *For any cell* c *at any level in* $\mathcal{G}(\mathbf{v})$, *let*

$$H(c)=\big\{\{i,j\}\mid 1\leq i\neq j\leq b \text{ and } i,j\in\mathcal{I}(c)\big\}.$$

Then $|\cup_{c\in\mathcal{G}(\mathbf{v})}H(c)|<t^{3r+4}b$.

PROOF. Note that for any two input blocks $\mathbf{v}_i$ and $\mathbf{v}_j$, if a cell c depends on both blocks at any level $\ell\leq r$, then the same cell also depends on the two blocks at level $r+1$. Thus, it suffices to bound the union of all cells at the last level.

Fix any tape $j\in[t]$. Let C be an input dependency string of tape j at level $r+1$. From Proposition 3.5, C can be decomposed into a set S of t^{r+1} interleaved monotone sequences with the requirement of cell portions as stated. For any cell c on this tape, let $d(c)$ be the right boundary of the cell portion of c in C. For any sequence $s\in S$ and for some $i\in[b]$, we say that c *stands at the* i*-th stage* in s if the rightmost entry on the left of $d(c)$ that s takes in C has value i. Note that s is monotone.

Define a matrix $\mathcal{T}$ whose t^{r+1} rows are indexed by the sequences in S and whose columns are indexed by the cells in tape j as follows: $\mathcal{T}$ has a number of columns equal to the number of cells with nonempty input dependency, where the columns are placed in the same 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 $\mathbf{c}$ corresponding to a cell c, let

$$H'(\mathbf{c})=\{\{\mathcal{T}_{s,\mathbf{c}}\ ,\ \mathcal{T}_{s',\mathbf{c}}\}\mid s\neq s' \text{ and } \mathcal{T}_{s,\mathbf{c}}\neq\mathcal{T}_{s',\mathbf{c}}\}.$$

Firstly, we observe that $H(c)\subset H'(\mathbf{c})$. To see this, consider some $\{i,j\}\in H(c)$. Then $i\neq j\in[b]$ and $i,j\in\mathcal{I}(c)$. Therefore i and j both appears in the cell portion of c in C. Then clearly there must be some $s\in S$ that takes i and some $s'\in S$ that takes j. Because of the property guaranteed by Proposition 3.5, $s\neq s'$. Thus, c stands at stage i in s and at stage j in s'. It follows that $\{i,j\}\in H'(\mathbf{c})$.

We also immediately get $|H'(\mathbf{c})|<(t^{r+1})^2=t^{2r+2}$.

For any two adjacent columns $\mathbf{c}$ and $\mathbf{c}'$ corresponding to two adjacent cells c and c', respectively, if $H'(\mathbf{c})\neq H'(\mathbf{c}')$, then there must be a 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 for any single s. Since there are at most t^{r+1} sequences, there are at most bt^{r+1} different $H'(\mathbf{c})$ over all columns. Thus $|\cup_{c\in\mathcal{G}(\mathbf{v})}H(c)|\leq t\cdot t^{2r+2}\cdot bt^{r+1}=bt^{3r+4}$, where the first factor "t" accounts for t different tapes. □

The next lemma specifically concerns functions of the form f^g_Φ such as $\textsc{pDisj}^{m,\Phi}_{n,p}$. For such a function, we define the following notation for the set of input coordinates for an f fed into g.

Definition 3.7. Given $\phi_1,\ldots,\phi_p$ permutations on $[m]$, and $i\in[m]$, define

$$J_i=\{(1,\phi_1(i)),(2,\phi_2(i)),\ldots,(p,\phi_p(i))\}.$$

Recall that r is the number of reversals in the algorithm. The following lemma intuitively says that for functions like f^g_Φ, if r is small, then there are many copies of f in f^g_Φ for which the algorithm cannot compare two input bits of f directly, as it does to solve the un-permuted promise set-disjointness problem, as noted in Section 2. A similar observation for the special case $p=2$ was made in [Grohe et al. 2006; 2009].

LEMMA 3.8. *Suppose that an input* $\mathbf{v}$ *is partitioned into* $b = p \cdot m$ *blocks:* $\mathbf{v} = (\mathbf{v}_{1,1}, \ldots, \mathbf{v}_{p,m})$. *For any sequence of permutations* $\phi_1, \phi_2, \ldots, \phi_p$ *on* $[m]$, *there exists a set* $I \subset [m]$ *with*

$$|I| \geq m - p \cdot t^{5r+8}\mathsf{relsorted}(\phi_1, \ldots, \phi_p)$$

such that for every $i \in I$, *there is no cell* c *in* $\mathcal{G}(\mathbf{v})$ *such that* $|J_i \cap \mathcal{I}(c)| > 1$, *where* J_i *is defined in Definition 3.7.*

PROOF. This is a generalization of an argument in [Grohe et al. 2006; 2009] which gave a proof for the special case $p = 2$. As in the previous proposition, it suffices to prove the claim for level $r + 1$. Let $Q = \{i \in [m] \mid \exists c : |J_i \cap \mathcal{I}(c)| > 1\}$. Then $I = [m] \setminus Q$. Thus, we need to prove that $|Q| \leq p \cdot t^{5r+8}\mathsf{relsorted}(\{\phi_1, \ldots, \phi_p\})$.

We fix any partition of Q into disjoint subsets $Q = \bigcup_{1 \leq p_1 < p_2 \leq p} Q_{p_1,p_2}$ such that

$$Q_{p_1,p_2} \subseteq \{i \in Q \mid \exists c : (p_1, \phi_{p_1}(i)), (p_2, \phi_{p_2}(i)) \in \mathcal{I}(c)\}$$

for any $1 \leq p_1 < p_2 \leq p$. Applying Proposition 3.6 (with $b = p$), 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 $|Q_{p_1,p_2}| \geq |Q|/\big(t^{3r+4}p\big)$. Fix these p_1 and p_2. We will next show that

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

which gives the claim.

We proceed to prove the last inequality; so, we focus on the two big chunks of input, $\mathbf{v}_{p_1,1}, \ldots, \mathbf{v}_{p_1,m}$ and $\mathbf{v}_{p_2,1}, \ldots, \mathbf{v}_{p_2,m}$, and argue that $\mathsf{sortedness}(\phi_{p_1}, \phi_{p_2})$ must be large if there are many i's (those in Q_{p_1,p_2}) for which the algorithm can simultaneously examine both $\mathbf{v}_{p_1,\phi_{p_1}(i)}$ and $\mathbf{v}_{p_2,\phi_{p_2}(i)}$.

Let $Q_{p_1,p_2} = \{i_1, \ldots, i_q\}$. Let $C \in \{(1,1), \ldots, (p,m)\}^*$ be the string obtained by concatenating the input dependency of all t tapes (at level $r + 1$), so that for any cell c, if $\mathcal{I}(c)$ contains both $(p_1, \phi_{p_1}(i))$ and $(p_2, \phi_{p_2}(i))$ for some $i \in Q_{p_1,p_2}$, then both of them are placed consecutively in C and in this order. Then there exists a permutation π on $[q]$ so that the following sequence

$$L = (p_1, \phi_{p_1}(i_{\pi(1)})), (p_2, \phi_{p_2}(i_{\pi(1)})), \ldots, (p_1, \phi_{p_1}(i_{\pi(q)})), (p_2, \phi_{p_2}(i_{\pi(q)}))$$

is a subsequence of C. Without loss of generality, for convenience of notation, assume that π is the identity permutation.

By Proposition 3.5, C, and hence L, can be decomposed into a set S of at most $t \cdot t^{r+1} = t^{r+2}$ monotone sequences. Thus there is some *monotone* sequence $\mathbf{s} \in S$ such that $\mathbf{s}$ contains at least $q/|S| \geq q/t^{r+2}$ entries of the form $(p_1, *)$ in L. 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 $j_1 < j_2 \in Q_1$, either

$$(\dagger) \qquad \phi_{p_1}(i_{j_1}) < \phi_{p_1}(i_{j_2}) \quad \text{or} \quad \phi_{p_1}(i_{j_1}) > \phi_{p_1}(i_{j_2}),$$

depending only on whether $\mathbf{s}$ is increasing or decreasing, respectively. Let the indices in Q_1 be $j_1 < \ldots < j_{q_1}$. Consider the following subsequence of L,

$$L' = (p_2, \phi_{p_2}(i_{j_1})), \ldots, (p_2, \phi_{p_2}(i_{j_{q_1}})).$$

As before, there must be at least one *monotone* sequence $\mathbf{s}' \in S$ such that there are at least $q_1/|S| = q_1/t^{r+2}$ such entries in $\mathbf{s}'$. In other words, there exists a set $Q_2 \subseteq Q_1$ of size at least $q_1/t^{r+2} \geq q/t^{2r+4}$ such that for every $l_1 < l_2 \in Q_2$, either $\phi_{p_2}(i_{l_1}) < \phi_{p_2}(i_{l_2})$ or $\phi_{p_2}(i_{l_1}) > \phi_{p_2}(i_{l_2})$, depending only on whether $\mathbf{s}'$ is increasing or decreasing, respectively. This fact together with (†) above gives us $\mathsf{sortedness}(\phi_{p_1}, \phi_{p_2}) \geq q/t^{2r+4}$, as required. □

4. SIMULATION OF READ/WRITE STREAM ALGORITHMS BY COMMUNICATION PROTOCOLS

In this section we apply the properties of dependency graphs and information flow in read/write stream algorithms in Section 3 to obtain a simulation of read/write stream algorithms for functions of the form f^g_Φ by communication protocols.

We will show that any given algorithm solving f^g_Φ can be reduced to a communication protocol solving the following p-party NIH communication game: (1) the p players are to compute f^g_Φ which contains m instances of f on distinct inputs, (2) except for some i-th instance of f, all of them already know the input to all the other f's, and (3) the p input blocks to the i-th instance of f are distributed to the p players. We will formally define this game shortly in Theorem 4.2. The key observation for this reduction is Lemma 3.8. Recall that this lemma intuitively says that for functions of the form f^g_Φ for "random" Φ, if the number of passes in an algorithm is small, then for most instances of f in f^g_Φ, those corresponding to the set I in Lemma 3.8, the contents written on any tape at any time in the algorithm depends only on the state (memory) of the algorithm and on at most 1 out of p input blocks to the instance of f. Thus, if the i-th instance of f in the above communication game is in I, the players can intuitively simulate the given algorithm for f^g_Φ by communicating the state of the algorithm to each other as necessary. If the algorithm uses small space and small number of passes, the communication cost is small.

We next define the construction of the input of f^g_Φ in the above communication game. In the following definition, it will be helpful to think of $\mathcal{Y}$ as the input blocks of the i-th instance of f that will be distributed to the players in the above communication game.

Definition 4.1. Fix $p \geq 2, m \geq 1, i \in [m]$, and $\Phi = (\phi_1, \ldots, \phi_p)$ to be a sequence of permutations on $[m]$. Let J_i be defined as in Definition 3.7. Let X be an input domain and $\mathcal{Y} = (Y_1, \ldots, Y_p)$ be an input in X^p. For each $\rho \in X^{p\times(m-1)}$, we define $\mathbf{v} = \mathbf{v}(\mathcal{Y}, i, \rho, \Phi) = (\mathbf{v}_{1,1}, \ldots, \mathbf{v}_{p,m}) \in X^{pm}$ such that $\mathbf{v}_{j,\phi_j(i)} = Y_j$ for every $j \in [p]$, and $\mathbf{v}_{|(\{(1,1),\ldots,(p,m)\}-J_i)} = \rho$.

The input $\mathbf{v}$ for f^g_Φ is constructed by using $\mathcal{Y}$ as an input for the i-th instance of f and using ρ as input for all other instances. The following is the main theorem of this section.

THEOREM 4.2. *For every integer $t > 0$ there is a constant $c > 0$ such that the following holds. Let $p \geq 2, m \geq 1, i \in [m]$, $\Phi = (\phi_1, \ldots, \phi_p)$ be any sequence of p permutations on $[m]$, X be any input domain with $n = \lceil \log_2(X) \rceil$, and $\rho \in X^{p\times(m-1)}$. Also let A be any deterministic (r, s, t)-read/write stream algorithm on $X^{p\times m}$ (which computes some function f^g_Φ). Then there is a p-party NIH protocol $P_{i,\rho,\Phi}$, in which all players know i, ρ, and Φ, with the following properties: on every input $\mathcal{Y} = (Y_1, \ldots, Y_p) \in X^p$ where each player j receives Y_j,*

(a) $|P_{i,\rho,\Phi}| \leq ct^r sr^2 p \log_2(pmn)$,
(b) if the protocol $P_{i,\rho,\Phi}$ does not "fail" on $\mathcal{Y}$, then $P_{i,\rho,\Phi}(\mathcal{Y}) = \mathsf{A}(\mathbf{v})$, where $\mathbf{v} = \mathbf{v}(\mathcal{Y}, i, \rho, \Phi)$ is defined as in Definition 4.1,
(c) but the protocol may "fail" on $\mathcal{Y}$, i.e., $P_{i,\rho,\Phi}(\mathcal{Y}) =$ "fail", however, this happens only if $i \notin I_{\mathbf{v}}$, where $I_{\mathbf{v}} \subseteq [m]$ is the set as guaranteed by Lemma 3.8.

PROOF. We describe the protocol first and then analyze it. Let $\mathcal{G}(\mathbf{v})$ be the dependency graph on input $\mathbf{v} = \mathbf{v}(\mathcal{Y}, i, \rho, \Phi)$.

After constructing $\mathbf{v}$, each player executes A on $\mathbf{v}$. Note that all players know all of $\mathbf{v}$ except 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 $\mathcal{G}(\mathbf{v})$. Each keeps track of the level (the pass) in $\mathcal{G}(\mathbf{v})$ that A is currently working on and

the machine state of A. Specifically, for every tape cell at every level in $\mathcal{G}(\mathbf{v})$ 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 be computed by a specific player.

The cells of type (1) are those cells c such that $(j, \phi_j(i)) \notin \mathcal{I}(c)$ for every $j \in [p]$. 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 type-(1) cell 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.

The cells of type (2) are those that depend on some input held by a particular player. Consider a cell c such that $(j, \phi_j(i)) \in \mathcal{I}(c)$ 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 content depends on the inputs from more than one player. As soon as the simulation comes to such a cell, it will stop and the protocol will output "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 contents 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 a cell of type (2) and all currently scanned type-(2) cells depend on a single 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 and the algorithm does not make any direction 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 $\mathcal{G}(\mathbf{v})$, as being 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". By Lemma 3.8, if $i \in I_{\mathbf{v}}$, this case will never happen, so part (c) is proved.

When A terminates, the protocol will output exactly as A does, and so part (b) is proved.

It remains to compute the communication cost. To do so, we need to bound the cost of each communication and the number of times a communication occurs. We need the following easy proposition due to Grohe and Schweikardt [2005]; we give a proof at the end of this section for completeness.

PROPOSITION 4.3. *([Grohe and Schweikardt 2005; Grohe et al. 2009]) The total number of tape cells reached in a (r, s, t)-read/write stream algorithm* A *is at most $2^{(r+1)(s+1)}M$, where M is the input length.*

Given Proposition 4.3, the cost of one communication is $t \log_2(2^{(r+1)(s+1)} pmn) + s \leq \Delta(rs + \log_2 pmn)$, for some constant $\Delta = \Delta(t) > 0$. When one player communicates, one of the tape heads has either just reversed or just moved from a cell depending on her to another cell that does not. The former means that the algorithm comes to the next level, which happens at most $r + 1$ times. By Lemma 3.4 (with $b = p$), the latter occurs at most $t^{r+2} + t^{r+1} + \ldots + 1$ times for a single tape and single player. Summing over all tapes and all p players, this occurs at most pt^{r+4} times in total. Thus, the total communication is

$$\Delta(rs + \log_2 pmn)(r + 1 + pt^{r+4}) \leq ct^r sr^2 p \log_2(pmn)$$

for some constant $c = c(t) > 0$, which proves part (a) and completes the proof of the theorem. □

It remains to prove Proposition 4.3.

Proof of Proposition 4.3. We upper bound the number of cells reached by the number of non-blank cells left by A. The initial number of non-blank cells is clearly M. It suffices to prove that after each reversal, the number of non-blank cells is multiplied by at most 2^{s+1}.

Consider the interval between any two consecutive reversals. Notice that in this time, if any head reaches a new (blank) cell, it will keep reaching new cells, since it does not reverse. Without loss of generality, assume that the order in which the heads reach new cells is head t first, then head $t - 1$, and so on. Consider any step in the pass, at which the currently scanned symbols are $(b_1, \ldots, b_i, \square, \ldots, \square)$, where $b_j \in \{0, 1\}$ for every $j \in [i]$ and there are $t - i$ blanks ("$\square$"). Then, no matter how the heads $i + 1, \ldots, t$ move in the next several steps, all the heads $1, \ldots, i$ can only stand still for at most 2^s more steps, otherwise the algorithm would be in a same state twice and loop forever. For the same reason, the pass must end in at most 2^s steps after the algorithm reads $(\square, \ldots, \square)$ of t blanks. Thus, if K is the number of non-blank cells before the pass, in this pass the algorithm can make at most $(K + 1)2^s \leq K2^{s+1}$ new cells, where the "+1" accounts for the final all-blank configuration. □

5. BOUNDS FOR DISJOINTNESS AND FREQUENCY MOMENTS

Using Lemma 2.3 and the fact that $\textsc{pDisj}^{m,\Phi}_{n,p}$ is a special case of $\textsc{pDisj}^{\Pi}_{N,p}$ where $N = mn$, we will lower bound the frequency moment problems by giving lower bounds for the $\textsc{pDisj}^{m,\Phi}_{n,p}$ problem. This problem was previously studied in [Beame et al. 2007] for the special case $p = 2$; our bounds extend those in [Beame et al. 2007] for any $p \geq 2$ and also improve the bounds for $p = 2$. We will show how to use our simulation described in the last section to prove lower bounds for $\textsc{pDisj}^{m,\Phi}_{n,p}$ for read/write stream algorithms.

To show lower bounds for $\textsc{pDisj}^{m,\Phi}_{n,p}$ for read/write streams algorithms, we will reduce any algorithm solving $\textsc{pDisj}^{m,\Phi}_{n,p}$ to a p-party NIH communication protocol solving the set-intersection problem $\textsc{pDisj}_{n,p}$, and then apply known communication lower bounds for the latter problem. We next discuss our ideas to obtain this reduction. Let $f = \textsc{pDisj}_{n,p}$, recalling that $\textsc{pDisj}^{m,\Phi}_{n,p} = f^{\vee}_{\Phi}$. Given any read/write stream algorithm A solving $f^{\vee}_{\Phi}$, the reduction in Theorem 4.2 gives us, for any $i \in [m]$ and any ρ which is the input for all instances of f, except the i-th instance, a communication protocol $\mathrm{P}_{i,\rho,\Phi}$ that simulates A, in which the p input blocks of the i-th instance of f are distributed to the p players. To further reduce this protocol to a communication protocol for $f = \textsc{pDisj}_{n,p}$, we need to resolve two issues. First of all, the simulation outputs the value of A which computes $f^{\vee}_{\Phi}$, an OR of m instances of f, not the value of a single f. Second of all, the simulation may fail.

We resolve the first issue by choosing ρ so that the values of all other instances of f is 0. Thus the value of $f_\Phi^\vee$ is exactly the value of the i-th instance. The fix for the second issue is more subtle. Theorem 4.2 tells us that the simulation fails only if the i-th instance does not belong to a "good" set $I_{\mathbf{v}}$, where $\mathbf{v}$ is an input to $f_\Phi^\vee$. Lemma 3.8 also tells us that if A does not make too many passes and Φ is some fixed set of random permutations, the set $I_{\mathbf{v}}$ is always large, say, containing at least a 3/4 fraction of all instances in $f_\Phi^\vee$. Thus, intuitively, we expect that if we choose $i \in [m]$ randomly, then with good probability, the i-th instance is in $I_{\mathbf{v}}$, and hence the simulation does not fail. However, the set $I_{\mathbf{v}}$, which is determined by A, depends on $\mathbf{v}$, and hence depends on all of i, ρ, and $\mathcal{Y}$, which is the input of the k players for the i-th instance of f. Thus, it may happen that i almost never belongs to $I_{\mathbf{v}}$. However, despite this dependency, we will construct $\mathbf{v}$ so that i and $I_{\mathbf{v}}$ are statistically independent, and thus, a randomly chosen i will belong in $I_{\mathbf{v}}$ with a good probability. This is the point that we need to use the special properties of the input of $f = \textsc{pDisj}_{n,p}$. Intuitively, we show that if $\mathcal{Y}$ is an input where $f(\mathcal{Y}) = 0$, we can choose ρ so that the inputs to the i-th instance of f and to all other instances look statistically the same. Hence A cannot tell which instance is the embedded i-th instance, and therefore $I_{\mathbf{v}}$ must be independent from i.

We proceed to describe the reduction formally.

LEMMA 5.1. *For any constant $t \geq 1$ there is a constant $c = c(t) > 0$ such that, given any randomized (r, s, t)-read/write stream algorithm A for $\textsc{pDisj}_{n,p}^{m,\Phi}$ on X^{pm} with error at most δ, where $X = \{0,1\}^n$ and $\frac{pt^{5r+8}\mathsf{relsorted}(\Phi)}{m} = d < 1$, there is a randomized p-party NIH protocol $\mathcal{P}$ for $\textsc{pDisj}_{n,p}$ of cost $ct^r sr^2 p \log_2(pmn)$ 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 its source of randomness. We denote by $\binom{[n]}{\ell}$ the set of $x \in X = \{0,1\}^n$ with $|x| = \ell$.

Let $f = \textsc{pDisj}_{n,p}$. Thus, $f^{-1}(0)$ is the set of of inputs each consisting of p pair-wise disjoint subsets of $[n]$. We also have that by definition, $\textsc{pDisj}_{n,p}^{m,\Phi} = f_\Phi^\vee$.

Next we describe the randomized communication protocol $\mathcal{P}$ for $\textsc{pDisj}_{n,p}$. On input $\mathcal{Y} = \{Y_1, \ldots, Y_p\} \in X^p$:

(1) Each player j broadcasts $|Y_j|$.
(2) The players use the public random bits to uniformly and randomly generate the following:
   (a) $\rho \in (f^{-1}(0) \cap [\times_{j=1}^p \binom{[n]}{|Y_j|}])^{m-1} \subseteq X^{p(m-1)}$,
   (b) $i \in [m]$,
   (c) $\mathfrak{R} \in \{0,1\}^\Gamma$,
   (d) a permutation $\tau : [n] \to [n]$.
(3) Each player j computes $Y_j' = \tau(Y_j)$ which, thinking of Y_j as subset of $[n]$, is a new subset obtained by applying τ to each element in Y_j.
(4) Let $\mathcal{Y}' = \{Y_1', \ldots, Y_p'\}$. The players run protocol $P_{i,\rho,\Phi}$ with input $\mathcal{Y}'$ obtained by applying Theorem 4.2 with $\mathsf{A}_\mathfrak{R}$ and $\mathbf{v} = \mathbf{v}(\mathcal{Y}', i, \rho, \Phi)$.
(5) If $P_{i,\rho,\Phi}$ outputs "fail" then output 1; else output what $P_{i,\rho,\Phi}$ does.

We analyze the error of this protocol $\mathcal{P}$. Let $\mathcal{G}(\mathbf{v})$ be the dependency graph induced by $\mathsf{A}_\mathfrak{R}$ on input $\mathbf{v}$ and $I_{\mathbf{v}} \subseteq [m]$ be the set defined by $\mathcal{G}(\mathbf{v})$ as guaranteed in Lemma 3.8. The correctness of the protocol depends on whether $i \in I_{\mathbf{v}}$ and whether $\mathsf{A}_\mathfrak{R}(\mathbf{v})$ is correct.

First we consider the case $Y_1, \ldots, Y_p$ are disjoint. After being re-mapped by τ, the sets $Y_1', \ldots, Y_p'$ are also disjoint and furthermore, uniformly distributed in $f^{-1}(0) \cap [\times_{j=1}^p \binom{[n]}{|Y_j|}]$. Hence, by construction, $\mathbf{v}$ is uniformly distributed over $(f^{-1}(0) \cap [\times_{j=1}^p \binom{[n]}{|Y_j|}])^m$. Therefore

$I_{\mathbf{v}}$ is statistically independent from i. By Lemma 3.8, $|I_{\mathbf{v}}| \geq (1-d)m$. Thus, the probability that $i \in I_{\mathbf{v}}$ is at least $1-d$. Formally, we have

$$\begin{aligned}\Pr[\mathcal{P}(\mathcal{Y}) = 0 \mid f(\mathcal{Y}) = 0] &\geq \Pr[\mathsf{A}_{\mathfrak{R}}(\mathbf{v}) = 0 \mid f(\mathcal{Y}) = 0] \\ &\quad \times \Pr[\mathcal{P}(\mathcal{Y}) = 0 \mid \mathsf{A}_{\mathfrak{R}}(\mathbf{v}) = 0, f(\mathcal{Y}) = 0] \\ &\geq (1-\delta)\Pr[\mathcal{P}(\mathcal{Y}) = 0 \mid \mathsf{A}_{\mathfrak{R}}(\mathbf{v}) = 0, f(\mathcal{Y}) = 0] \\ &\geq (1-\delta)\Pr[i \in I_{\mathbf{v}} \mid \mathsf{A}_{\mathfrak{R}}(\mathbf{v}) = 0, f(\mathcal{Y}) = 0] \\ &\geq (1-\delta)(1-d).\end{aligned}$$

Next we consider the case that the sets $Y_1, \ldots, Y_p$ are not disjoint, i.e., $f(\mathcal{Y}) = 1$. Thus, by construction, $f(\mathcal{Y}') = 1$ and $f^{\vee}_{\Phi}(\mathbf{v}) = 1$. In this case if $P_{i,\rho,\Phi}(\mathcal{Y}')$ outputs "fail", the protocol always outputs correctly. Otherwise it will output what A does. In other words,

$$\begin{aligned}\Pr[\mathcal{P}(\mathcal{Y}) = 1 \mid f(\mathcal{Y}) = 1] &= \Pr[P_{i,\rho,\Phi}(\mathcal{Y}') = fail \mid f(\mathcal{Y}) = 1] \\ &\quad + \Pr[P_{i,\rho,\Phi}(\mathcal{Y}') = 1 \mid f(\mathcal{Y}) = 1] \\ &= \Pr[P_{i,\rho,\Phi}(\mathcal{Y}') = fail \mid f(\mathcal{Y}) = 1] \\ &\quad + \Pr[\mathsf{A}_{\mathfrak{R}}(\mathbf{v}) = 1 \text{ and } P_{i,\rho,\Phi}(\mathcal{Y}') \neq fail \mid f(\mathcal{Y}) = 1] \\ &\geq \Pr[\mathsf{A}_{\mathfrak{R}}(\mathbf{v}) = 1 \mid f(\mathcal{Y}) = 1] \\ &= \Pr[\mathsf{A}_{\mathfrak{R}}(\mathbf{v}) = 1 \mid f^{\vee}_{\Phi}(\mathbf{v}) = 1] \geq 1-\delta,\end{aligned}$$

where the last inequality follows from the fact that $\Pr[B] + \Pr[A \wedge \neg B] \geq \Pr[A \wedge B] + \Pr[A \wedge \neg B] = \Pr[A]$ for any probability events A and B.

Thus the bound on the error follows. Finally, the cost of $\mathcal{P}$ is $p \log_2 n$ plus the cost of $P_{i,\rho,\Phi}$, which completes the lemma. □

We are ready to derive our read/write streams lower bound for $\textsc{pDisj}^{m,\Phi}_{n,p}$.

Lemma 5.2. *Let $1/4 > \delta \geq 0$, $1/2 > \beta > 0$, $1 > \eta > \alpha > 0$, and $t \in \mathbb{N}$ be any constants. Then for any $r, s : \mathbb{N} \mapsto \mathbb{R}^+$ such that $r(N) = o(\log N)$ and $s(N) = o(N^{1-\frac{4\beta}{2\beta+1}-\eta})$, there is no randomized $\big(r, s, t\big)$-read/write stream algorithm with error at most δ for $\textsc{pDisj}^{m,\Phi^*}_{n,p}$, where $p \leq m^{\frac{1}{2}-\alpha} = n^{\beta}, N = mn$, and $\Phi^* = \Phi^*_{p,m}$ as guaranteed in Corollary 2.2.*

Proof. Suppose for contradiction that there is such an algorithm A. Then by Lemma 5.1, there is a randomized NIH communication protocol for $\textsc{pDisj}_{n,p}$ with cost $O(t^r s r^2 p \log_2(pmn))$ and with error at most $d + (1-d)\delta$, where $d = \frac{pt^{5r+8}\mathsf{relsorted}(\Phi^*)}{m}$. For any constant $a > 0$ and N sufficiently large, we have $r(N) \leq a \log_2 N = a(1 + \frac{1-2\alpha}{2\beta}) \log_2 m$. Since $\mathsf{relsorted}(\Phi^*) \leq 2e\sqrt{m}$, we have $d + (1-d)\delta \leq 2\delta < 1/2$ for a sufficiently small depending on δ, t, and α.

By [Gronemeier 2009; Jayram 2009], this communication complexity must be $\Omega(\frac{n}{p})$. This gives us, for some suitable constant η' depending on a, η, and α that $s = \Omega(n^{1-2\beta-\eta'})$ which is $\Omega(N^{\frac{(1-2\beta-\eta')(1-2\alpha)}{1+2\beta-2\alpha}})$ and hence $\Omega(N^{1-\frac{4\beta}{2\beta+1}-\eta})$, which violates our assumption about A. The lemma follows. □

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

Theorem 5.3. *Let $1/4 > \delta, \gamma > 0$, $\eta > 0$, and $t \geq 1$ be constants. Then there exists an infinite family $\{\Pi^*_{p,N}\}$, where each $\Pi^*_{p,N}$ is a sequence of p permutations on $[N]$ and $p \leq N^{\gamma}$, such that there is no randomized $\big(o(\log N), N^{1-4\gamma-\eta}, t\big)$-read/write stream algorithm with error at most δ for $\textsc{pDisj}^{\Pi^*}_{N,p}$.*

PROOF. This follows from Lemma 5.2 where we set $N = mn$. Let $\Phi^*_{p,m}$ be the sequence of permutations on $[m]$ used in Lemma 5.2. Let $\Pi^*_{p,N}$ be the extension of $\Phi^*_{p,m}$ to p permutations on $[N]$ as described in Section 2. The theorem follows by applying the lemma with $\alpha, \beta > 0$ 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 $k > 1$, $t \geq 1$, $\eta > 0$ be constants. 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 every input stream of up to N elements from $[N]$, for any $1 \geq \epsilon \geq 1/N$.*

PROOF. Let ζ be such that $\epsilon = N^{-\zeta}/4$. Then by setting $p = \lceil N^{(1+\zeta)/k} \rceil = \lceil (4\epsilon N)^{1/k} \rceil$, Lemma 2.3(a) and Theorem 5.3 imply that no randomized $(o(\log N), O(N^{1-4(1-\zeta)/k-\eta}), t)$-read/write stream algorithm can compute F_k within a $(1+\epsilon)$ factor. 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 $\eta > 0$ be constants. 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 every input stream of up to N elements from $[N]$.*

With the same proof and by interpreting $1/\infty = 0$, we derive new lower bound for approximating F^*_∞.

COROLLARY 5.6. *For any constants $t \geq 1$ and $\eta > 0$, there cannot exist any 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 any 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 $1 > k \geq 0$, $t \geq 1$, $\eta > 0$ be constants. 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 any input stream of up to N elements from $[N]$, where $1/N^{3/4+\eta} > \epsilon \geq 1/N$.*

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

5.1. An upper bound for $\text{pDISJ}^{m,\Phi}_{n,p}$ and a combinatorial property of a set of permutations

Our lower bound for $\text{pDISJ}^{\Pi}_{N,p}$ is only interesting when $N = \omega(p^4)$. This is because in order for the reduction from $\text{pDISJ}^{m,\Phi}_{n,p}$ 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 $\text{pDISJ}_{n,p}$, which is optimal. Lemma 5.8 will show that a condition requiring m to be polynomially larger than p is also necessary, and thus an approach that relies on $\text{pDISJ}^{m,\Phi}_{n,p}$ cannot yield lower bounds for constant factor approximations of F_k for $k < 3.5$.

LEMMA 5.8. *For any integers $m < p^{3/2}/64$ and 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* $\text{PDISJ}_{n,p}^{m,\Phi}$.

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. [Steele 1995]) 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 $< m$ which is a contradiction. □

It is not hard to show that the above lemma is tight, even for any four permutations. As an example, a set of 4 permutations on [8] in which no pair has a common subsequence of length longer than 2 is the following: $(1, 2, 3, 4, 5, 6, 7, 8)$, $(4, 3, 2, 1, 8, 7, 6, 5)$, $(7, 8, 5, 6, 3, 4, 1, 2)$, and $(6, 5, 8, 7, 2, 1, 4, 3)$. This can be generalized to any $[m]$: the first is the identity permutation; the second is an increasing sequence of $m^{1/3}$ decreasing subsequences, each of length $m^{2/3}$; the third is a decreasing sequence of $m^{2/3}$ increasing sequences, each of length $m^{1/3}$; and the fourth is a suitably alternating sequence of $m^{2/3}$ decreasing sequences, each of length $m^{1/3}$.

PROOF OF LEMMA 5.8. Given Φ there exist $L_1, L_2, \ldots, L_{p/3}$ defined as follows: L_1 is a common subsequence of two of ϕ_1, ϕ_2, and ϕ_3, of length at least $m^{1/3}$ given by Lemma 5.9; L_2 is a common subsequence of two of ϕ_4, ϕ_5, and ϕ_6 that is disjoint from L_1 and of length at least $(m - |L_1|)^{1/3}$; L_3 is a common subsequence of two of ϕ_7, ϕ_8, and ϕ_9 disjoint from $L_1 \cup L_2$ and of length at least $(m - |L_1| - |L_2|)^{1/3}$, and so on, with no index $i \in [m]$ appearing in more than one sequence. For each of the L_j let a_j and b_j denote the indices of the two permutations having the common subsequence L_j. The number of elements that do not appear in $L_1, \ldots, L_\ell$ 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. Thus $m_{p/8+p/32} \leq m/64$. In general we have that for any $j \geq 1$,

$$m_{p/2^3 + p/2^5 + \ldots + p/2^{2j+1}} \leq m/8^j.$$

Thus $m_{p/3} = 0$, which implies that every $i \in [m]$ is in exactly one L 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 $L_{p/3}, \ldots, L_2, L_1$, 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 L_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 $L_{p/3}, \ldots, L_1$. These sequences can be hardwired into the state transition function as follows. We represent the sequence of positions indicated by $L_{p/3}, \ldots, L_1$ 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 L_j is being considered, $i \leq \lceil \sqrt{m} \rceil$ denotes the position within the common subsequence L_j that is being considered, a denotes the index of one of the two permutations that have L_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 L_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 $i+1 \leq |L_j|$ 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 number of bits of state required is $O(\log pmn)$. □

6. OPEN QUESTIONS

The general question that we have begun to answer here is: for what natural approximation problems does the read/write streams model (with $o(\log N)$ passes) add significant power to the data stream model? We have mostly but not fully resolved the case of frequency moments – can one 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, $\textsc{pDisj}^{m,\Phi}_{n,p}$. Thus we expect that improvements in the lower bound, if any, will require working with more general instances of $\textsc{pDisj}^{\Pi}_{N,p}$ than those provided by $\textsc{pDisj}^{m,\Phi}_{n,p}$. However, it is not clear how our simulation given in Theorem 4.2 can be made to work for more general instances of $\textsc{pDisj}^{\Pi}_{N,p}$. On the other hand, we believe that our simulation can be used or extended to derive interesting lower bounds for other natural functions, and leave this direction for future work.

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 originally conjectured that subsequences of length $m^{1/2-o(1)}$ must exist for $k = O(\log m)$, which would have shown the approximate optimality of our algorithm, but recently $\tilde{O}(m^{1/3})$ upper bounds have been shown for this case [Beame et al. 2009].

ACKNOWLEDGMENTS

We would like to thank T.S. Jayram and Atri Rudra for useful discussions and anonymous referees for very helpful comments.

REFERENCES

Alon, N., Matias, Y., and Szegedy, M. 1999. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences 58,* 1, 137–147.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. 2002. Models and issues in data stream systems. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Database Systems*. 1–16.

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

Beame, P., Blais, E., and Huynh-Ngoc, D.-T. 2009. Longest common subsequences in sets of permutations. Tech. Rep. arXiv:R0904.1615v1 [math.CO], arxiv.

Beame, P. and Huynh-Ngoc, D.-T. 2008. On the value of multiple read/write streams for approximating frequency moments. In *Proceedings 49th Annual Symposium on Foundations of Computer Science*. IEEE, Philadelphia, PA, 499–508.

Beame, P., Jayram, T. S., and Rudra, A. 2007. Lower bounds for randomized read/write stream algorithms. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*. San Diego, CA, 689–698.

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

Chakrabarti, A., Khot, S., and Sun, X. 2003. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *Proceedings Eighteenth Annual IEEE Conference on Computational Complexity*. Aarhus, Denmark, 107–117.

Chakrabarty, A. and Regev, O. 2011. An optimal lower bound on the communication complexity of gap-hamming-distance. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*. San Jose, CA, 51–60.

Grohe, M., Hernich, A., and Schweikardt, N. 2006. Randomized computations on large data sets: Tight lower bounds. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Database Systems*. 243–252.

Grohe, M., Hernich, A., and Schweikardt, N. 2009. Lower bounds for processing data with few random accesses to external memory. *Journal of the ACM 56,* 3, 1–58.

Grohe, M. and Schweikardt, N. 2005. Lower bounds for sorting with few random accesses to external memory. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Database Systems*. 238–249.

Gronemeier, A. 2009. Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness. In *(STACS) 2009: 26th Annual Symposium on Theoretical Aspects of Computer Science*. LIPICS. Schloss Dagstuhl – Leibniz Center for Informatics, Freiburg, Germany, 505–516.

Indyk, P. and Woodruff, D. 2003. Tight lower bounds for the distinct elements problem. In *Proceedings 44th Annual Symposium on Foundations of Computer Science*. IEEE, Boston, MA, 283–292.

Indyk, P. and Woodruff, D. P. 2005. Optimal approximations of frequency moments of data streams. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. Chicago, IL, 202–208.

Jayram, T. S. 2009. Hellinger strikes back: A note on the multi-party information complexity of and. In *APPROX-RANDOM*. Lecture Notes in Computer Science Series, vol. 5687. Springer-Verlag, Berkeley, CA, 562–573.

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

Razborov, A. A. 1992. On the distributional complexity of disjointness. *Theoretical Computer Science 106,* 2, 385–390.

Saks, M. E. and Sun, X. 2002. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*. Montreal, Quebec, Canada, 360–369.

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

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

Woodruff, D. 2004. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. 167–175.