Abstract—We prove a time-space tradeoff lower bound of $T = \Omega\left(n \log\left(\frac{n}{S}\right) \log \log\left(\frac{n}{S}\right)\right)$ for randomized oblivious branching programs to compute $1\text{GAP}$, also known as the pointer jumping problem, a problem for which there is a simple deterministic time $n$ and space $O(\log n)$ RAM (random access machine) algorithm. We give a similar time-space tradeoff of $T = \Omega\left(n \log\left(\frac{n}{S}\right) \log \log\left(\frac{n}{S}\right)\right)$ for Boolean randomized oblivious branching programs computing $GIP$-$MAP$, a variation of the generalized inner product problem that can be computed in time $n$ and space $O(\log^2 n)$ by a deterministic Boolean branching program.

These are also the first lower bounds for randomized oblivious branching programs computing explicit functions that apply for $T = \omega(n \log n)$. They also show that any simulation of general branching programs by randomized oblivious ones requires either a superlogarithmic increase in time or an exponential increase in space.

Keywords—time-space tradeoffs, lower bounds, branching programs, oblivious computation, randomization

I. INTRODUCTION

An algorithm is oblivious (sometimes also called input-oblivious) if and only if its every operation, operand, as well as the order of those operations is determined independent of its input. Certain models of computation, such as circuits or straight-line programs are inherently oblivious. However, many computing models such as Turing machines and random access machines (RAMs), which use non-oblivious operations such as indirect addressing, are not, though fairly efficient simulations of these general models by their more restricted oblivious variants have been shown [20], [3].

Our main result implies that a superlogarithmic increase in time or an exponential increase in space is necessary to convert a general algorithm to a randomized oblivious one. We derive this separation by considering a very simple problem for deterministic RAM algorithms, the pointer jumping problem of out-degree 1 graph reachability, $1\text{GAP}_n$.∗

∗Research supported by NSF grant CCF-0830626. Part of this research was done at the Institute for Advanced Study and also supported by NSF grant CCF-0832797.

Our lower bounds apply not only to randomized oblivious RAM algorithms but also to more powerful randomized oblivious branching programs. Branching programs are the natural generalization of decision trees to directed acyclic graphs and simultaneously model time and space for both Turing machines and RAMs: Time is the length of the longest path from the start (source) node to a sink and space is the logarithm of the number of nodes. Our precise results are the following.

Theorem 1. Let $\epsilon < 1/2$. Randomized oblivious branching programs or random access machines computing $1\text{GAP}_n$ using time $T$, space $S$ and with error at most $\epsilon$ require $T = \Omega\left(n \log\left(\frac{n}{S}\right) \log \log\left(\frac{n}{S}\right)\right)$.

Since $1\text{GAP}_n$ can be computed by a RAM algorithm in time $n$ and space $O(\log n)$, which follows the path from vertex 1 maintaining the current vertex and a step counter, we immediately obtain the following corollary.

Corollary 2. Any method for converting deterministic random access machine algorithms to randomized oblivious algorithms requires either an $n^{1-o(1)}$ factor increase in space or an $\Omega(\log n \log \log n)$ factor increase in time.

The $1\text{GAP}_n$ problem has input variables from a linear-sized domain that the RAM can read in one step. Because of this, the lower bound for computing $1\text{GAP}_n$ is at most $\Omega(\log \log (n/S))$ larger than the number of its input bits and so is sub-logarithmic for Boolean branching programs. However, we also obtain analogues of the above results for Boolean branching programs computing a variant of the generalized inner product problem that we denote by $GIP$-$MAP$.

Deterministic oblivious branching programs have been studied in many contexts. Indeed the much-studied ordered binary decision diagrams (or OBDDs) [11] correspond to the special case of deterministic oblivious branching programs that are also read-once in that any source–sink path queries each input at most once. Our lower bound approach follows a line of
work based on another reason to consider oblivious algorithms: their behavior is restricted and thus simpler to analyze than that of general algorithms. Alon and Maass [5] showed $T = \Omega(n \log(n/S))$ lower bounds for certain natural Boolean functions on deterministic oblivious branching programs and this lower bound tradeoff was increased by Babai, Nisan, and Szegedy [6] to $T = \Omega(n \log^2(n/S))$ for a different Boolean function based on the generalized inner product. Beame and Vee [10] also used a variant of [6] using generalized inner product to give an $\Omega(\log^2 n)$ factor separation between general branching programs and deterministic oblivious branching programs by proving a $T = \Omega(n \log^2(n/S))$ lower bound for the $1\text{GAP}_n$ problem. However, that separation does not apply to the randomized simulations nor to the Boolean branching programs that we consider.

Though a number of time-space tradeoff lower bounds have been proven for natural problems in NP for general deterministic and nondeterministic [9], [1], [2] and randomized [8] computation, all of the lower bounds are sub-logarithmic and, naturally, none can yield a separation between general and randomized oblivious branching programs. Indeed the largest previous lower bounds for solving decision problems on randomized branching programs are of the form $T = \Omega(n \log(n/S))$ which is at most a logarithmic factor larger than the trivial time bound of $n$. These bounds also apply to randomized read-$k$ branching programs (which roughly generalize oblivious branching programs for related problems) for $k = O(\log n)$ [21]. No prior separations of randomized oblivious computations from general computation have been known.

Our argument uses the well-known connection between oblivious branching programs and (best-partition) communication complexity that is implicit in [5] and first made explicit in the context of multiparty communication complexity in [12], [6]. We make use of the fact that inputs to the $1\text{GAP}_n$ problem conveniently encode any function computable by a small branching program.

More precisely, we show that, since the generalized inner product $\text{GIP}$ can be computed by a constant-width read-once branching program, we can convert any oblivious branching programs for $1\text{GAP}_n$ to a (partially) oblivious branching program that computes Permutted-$\text{GIP}$ which takes as input both a $\text{GIP}$ input $z$ and a permutation $\pi$ to determine how the $\text{GIP}$ function is applied to $z$. The permutation allows us to convert the lower bound for $\text{GIP}$ in fixed-partition multiparty communication complexity to a best-partition lower bound, reminiscent of a similar conversion in the context of 2-party communication complexity [16].

Though that idea would have been sufficient for the deterministic non-Boolean separations in [10], we need much more here. The key to our argument is a way of extending this idea to a more involved analysis that yields a reduction that works for distributional complexity. Our Boolean lower bounds follow for $\text{GIP-MAP}$, a version of Permutted-$\text{GIP}$ based on pseudo-random permutations.

The questions we consider here were inspired by recent results of Ajtai [3], [4] (and similar results of Damgård, Meldgaard, and Nielsen [13]) who, eliminating cryptographic assumptions from [14], [19], [15], showed efficient simulations of general RAM algorithms by randomized algorithms that are oblivious and succeed with high probability, with only a polylogarithmic factor overhead in both time and space. However, our separations do not apply in the context of their simulations for two reasons. First, their simulations assume that the original RAM algorithm only has sequential access to its input in which case the small space upper bound for $1\text{GAP}_n$ does not apply (or alternatively the original RAM algorithm has linear space which a deterministic oblivious branching program can use to compute any function in linear time). Second, our lower bounds apply only to randomized oblivious simulations, in which the sequence of locations accessed must be independent of the input but may depend on the random choices, whereas Ajtai’s simulations are more general oblivious randomized simulations in that the probability distribution of the sequence of locations accessed is input independent$^1$.

2. Preliminaries and Definitions

**Branching programs:** Let $D$ be a finite set and $n$ a positive integer. A $D$-way branching program is a connected directed acyclic graph with special nodes: the source node, the 1-sink node and the 0-sink node, a set of $n$ inputs and one output. Each non-sink node is labeled with an input index and every edge is labeled with a symbol from $D$, which corresponds to the value of the input in the originating node. The branching program computes a function $f : D^n \to \{0, 1\}$ by starting at the source and then proceeding along the nodes of the graph by querying the corresponding inputs and following the corresponding edges. The output is the label of the sink node reached. The time $T$ of a...

$^1$A simple algorithm that flips a random bit $r$ and then chooses which order to read bits $x_2$ and $x_3$ based on the value of $x_1 \oplus r$ is oblivious randomized but not randomized oblivious: Each order of reading $x_2, x_3$ has probability 1/2 independent of the input, but for each fixed value of $r$, the order of reading $x_2, x_3$ depends on the value of $x_1$.
branching program is the length of the longest path from the source to a sink and the space \( S \) is the logarithm base 2 of the number of the nodes in the branching program. The branching program is Boolean if \( D = \{0, 1\} \).

A branching program \( B \) computes a function \( f \) if for every \( x \in D^n \), the output of \( B \) on \( x \), denoted \( B(x) \), is equal to \( f(x) \). \( B \) approximates \( f \) under \( \mu \) with error at most \( \epsilon \) if \( B(x) = f(x) \) for all but an \( \epsilon \)-measure of \( x \in D^n \) under distribution \( \mu \). (For a probability distribution \( \mu \) we write \( \text{supp}(\mu) \) denote its support.)

A branching program is leveled if the underlying graph is leveled. For a leveled branching program, the width is the maximum number of nodes on any of its levels and thus the space of a width \( W \) leveled branching program is at least \( \log W \). (We write \( \log x \) to denote \( \log_2 x \).) A branching program is read-once if each input is queried at most once on every path in the program. An oblivious branching program is a leveled branching program in which the same input symbol is queried by all the nodes at any given level. A randomized oblivious branching program \( B \) is a probability distribution over deterministic oblivious branching programs with the same input set. \( B \) computes a function \( f \) with error at most \( \epsilon \) if for every input \( x \in D^n \), \( \Pr_{x \sim B}[B(x) = f(x)] \geq 1 - \epsilon \).

For \( I \subseteq [n] \), we also find it useful to define the \( I \)-time, \( I \)-size, and \( I \)-width of a branching program to denote the previous measures where we only count nodes at which variables with indices in \( I \) are queried. A branching program is \( I \)-oblivious if it is leveled and any level at which a variable with an index in \( I \) is queried, the same variable is queried at each node in the level.

Multiparty communication complexity: Let \( f : D^n \rightarrow \{0, 1\} \). Assume that \( p \) parties, each having access to a part of the input \( x \), wish to communicate in order to compute \( f(x) \). The set \( [n] \) is partitioned into \( p \) pieces, \( \mathcal{P} = \{P_1, P_2, \ldots, P_p\} \) such that each party \( i \) has access to every input whose index in \( P_j \) for \( j \neq i \). (Because player \( i \) has access to all of the inputs except for those in set \( P_i \), we can view the set \( P_i \) as a set of inputs being written on player \( i \)’s forehead.) The parties communicate by broadcasting bits to the other players, which can be viewed as writing the bits on a common board, switching turns based on the content of the board. The number-on-forehead (NOF) multiparty communication complexity of \( f \) with respect to the partition \( \mathcal{P} \), denoted \( C_p(f) \), is the minimum total number of bits written. When there is a standard partition of the input into \( p \)-parties associated with a given function \( f \), we will simply write \( C_p(f) \) instead of \( C_{\mathcal{P}}(f) \).

Let \( \mu \) be a probability distribution over \( D^n \). The \((\mu, \epsilon)\)-distributional communication complexity of \( f \) with respect to \( \mathcal{P} \), denoted \( D_{\mu, \mathcal{P}}^\epsilon(f) \), is the minimum number of bits exchanged in a NOF communication protocol with input partition \( \mathcal{P} \) that computes \( f \) correctly on a \( 1 - \epsilon \) fraction of the inputs weighted according to \( \mu \). Again, we replace the \( \mathcal{P} \) by \( p \) when \( \mathcal{P} \) is a \( p \)-partition that is understood in the context.

Branching Program Complexity and NOF Multiparty communication: Let \( B \) be a deterministic \( I \)-oblivious branching program of width \( W \) computing a function from \( D^n \) to \( \{0, 1\} \). Since \( B \) is \( I \)-oblivious, it yields a sequence \( \pi \) of queries to the input symbols indexed by \( I \). The following proposition is adapted from [6], [10] for oblivious BPs approximating a function.

**Proposition 3.** Let \( f : D^n \rightarrow \{0, 1\} \). Let \( B \) be a deterministic \( I \)-oblivious branching program of width \( W \) that approximates \( f \) under a probability distribution \( \mu \) with error at most \( \epsilon \). Let \( \mathcal{P}' \) be a partition of a subset \( I' \subseteq I \) of \( [n] \), and let \( \mu \) be a distribution on \( D^n \) that has fixed values for all input variables indexed by \( [n] \setminus I' \). Suppose that the query sequence of \( B \) can be written as \( s_1, \ldots, s_t \) such that for each \( s_i \), there is some class \( P_{i_j} \in \mathcal{P}' \) whose variables do not appear in \( s_i \). Then for any partition \( \mathcal{P} \) that extends \( \mathcal{P}' \) to all of \( [n] \), \((r - 1) \log(W) + 1 \geq D_{\mu, \mathcal{P}}^\epsilon(f) \).

**Proof:** Associate party \( j \) with each class \( P_j \) of \( \mathcal{P} \) and place all input variables in class \( P_j \) on the forehead of party \( j \). For \( i = 1, \ldots, r \), party \( j_i \) simulates \( B \) on segment \( s_i \) and broadcasts the name of the node of \( B \) reached at the end of the segment.

**Pointer Jumping and Generalized Inner Product:** We consider the out-degree 1 directed graph reachability problem, \( 1\text{GAP} \), which is also known as the pointer jumping problem. Define \( 1\text{GAP}_n : [n + 1]^n \rightarrow \{0, 1\} \) such that \( 1\text{GAP}_n(x) = 1 \) iff there is a sequence of indices \( i_1, i_2, \ldots, i_{\ell} \) such that \( i_1 = 1 \), \( i_{\ell} = n + 1 \) and \( x_{i_j} = i_{j+1} \) for all \( j = 2, \ldots, \ell - 1 \). The \( 1\text{GAP}_n \) problem is \( s \)-t connectivity in \((n + 1)\)-vertex directed graphs of out-degree 1, where \( x_1, \ldots, x_n \) represent the out-edges of nodes 1 through \( n \), \( s \) is 1, and \( t \) is \( n + 1 \) and vertex \( n + 1 \) has a self-loop.

We will relate the complexity of \( 1\text{GAP}_n \) for randomized oblivious branching programs to that of the generalized inner product problem \( \text{GIP}_{p,n} \) for a suitable value of \( p \). \( \text{GIP}_{p,n} : ([0, 1]^p)^n \rightarrow \{0, 1\} \) is given by \( \text{GIP}_{p,n}(z_1, \ldots, z_p) = \bigoplus_{j=1}^n A_{i=1}^p z_{ij} \). The standard input partition for \( \text{GIP}_{p,n} \) places each \( z_i \) in a separate class. Babai, Nisan, and Szegedy [6] proved that under the uniform distribution the \( p \)-party NOF
communication complexity of $GIP_{p,n}$ is large. We also will use the fact that $GIP_{p,n}$ can be computed easily by a leveled width 4, oblivious read-once branching program.

3. REDUCING PERMUTED PROBLEMS TO $1GAP$

In order to derive time-space tradeoffs for the $1GAP$ problem, we will use a reduction from a permuted version of $GIP$. Since the idea is more general we state the reduction more generally.

Let $f : \{0, 1\}^N \rightarrow \{0, 1\}$ be a Boolean function. Define the promise problem $Permuted-f : \{0, 1\}^N \times [N]^N \rightarrow \{0, 1\}$ by $Permuted-f(z, \pi) = f(z_{\pi(1)}, z_{\pi(2)}, \ldots, z_{\pi(N)})$ where $\pi$ is guaranteed to be a permutation.\(^2\)

Lemma 4. Let $n = Nw + 1$. If $f : \{0, 1\}^N \rightarrow \{0, 1\}$ is a Boolean function computable by a width $w$ oblivious read-once branching program then there is a reduction $g$ from $Permuted-f$ to $1GAP_n$ mapping $(z, \pi)$ to $x$ such that the value of each $x_i$ depends on $\pi$ and at most one bit of $z$, whose index is independent of $\pi$.

Proof: Let $B_f$ be a width-$w$ oblivious read-once branching program computing $f$. We assume wlog that $B_f$ queries the bits of $z$ in the order $z_1, z_2, \ldots, z_N$; if not, we can modify the construction below by applying the fixed ordering given by the queries of $B_f$. Given $\pi$, the function $Permuted-f$ is computed by a modification of $B_f^\pi$ that replaces each query to $z_j$ in $B_f$ by a query to $z_{\pi(j)}$.

Vertex $n + 1$ for the $1GAP_n$ problem will correspond to the 1-sink of $B_f^\pi$. Vertex 1 will point to the start node of $B_f^\pi$ and will also be identified with the 0-sink node of $B_f^\pi$. More precisely, $x_1$ has value $w \ast (\pi(1) - 1) + 2$, assuming that the first node in that level is the start node.

Vertices 2 though $n$ will correspond to the nodes of $B_f^\pi$. For $j \in [N]$ and $k \in [w]$, vertex $i = (j - 1) \ast w + k + 1$ will correspond to the $k$-th node of the level of $B_f^\pi$ that queries $z_j$. Note that, given $i$, the values $j$ and $k$ are uniquely determined. More precisely, given $i \in [2, n]$, the value of $x_i$ is determined as follows: Determine $j$ and $k$. Query $z_j$. Also query\(^3\) $\pi$ to determine $\ell = \pi^{-1}(j)$, the level in $B_f^\pi$ at which $z_j$ is queried. Unless $\ell = N$, the next variable that $B_f^\pi$ will query is $z_{\pi(\ell+1)}$. Suppose that the 0- and 1-outedges from the $k$-th node at level $\ell$ in $B_f$ are to the $k_0$-th and $k_1$-th nodes at level $\ell + 1$ in $B_f$ respectively; then the value of $x_i$ will be $\pi(\ell+1) \ast w + k_{x_i} + 1$. Otherwise, set the value of $x_i$ depending on $z_j$ to either 1 or $n + 1$ depending on whether the corresponding edge goes to the 0-sink or 1-sink. Correctness is immediate. \(\blacksquare\)

Overloading notation we identify the subset $[N]$ of the $2N$ indices of $Permuted-f$ with $z$.

Corollary 5. Let $n = Nw + 1$ and $f : \{0, 1\}^N \rightarrow \{0, 1\}$ be a Boolean function computable by a width-$w$ oblivious read-once branching program. If there is a (randomized) oblivious branching program for $1GAP_n$ of time $T$, space $S$, and width $W$, then there is a (randomized) $z$-oblivious branching program for $Permuted-f$ with $z$-time $T$, $z$-space $S$, and $z$-width $W$.

Proof: Use the reduction $g$ given by Lemma 4 to replace each query to an input $x_i$ of $1GAP_n$ by the single query of $z_j$ depending on $i$ and then query $\pi$ as given by $g$. There is one $z$-query for each $x$-query. (We do not care that queries to $\pi$ are efficient.) \(\blacksquare\)

The construction shown in Figure 1 is easily generalized to show that we can apply Corollary 5 to $GIP_{p,m}$.

**Proposition 6.** Let $p, m \geq 0$ and $N = mp$. There is a width-4 read-once oblivious branching program for $GIP_{p,m}$.
4. Time-Space Tradeoff Lower Bound for Permutated-GIP and 1GAP

Following the last section, to derive lower bounds for 1GAP, we consider randomized $z$-oblivious branching programs for Permutated-GIP. Such programs are distributions over deterministic programs, as usual we invoke the easy half of Yao’s Lemma [22] to show that it is enough to find a distribution $\mu$ such that every $\epsilon$-error deterministic $z$-oblivious branching program approximating Permutated-GIP under $\mu$ will have a large time-space tradeoff.

The distribution $\mu$ we consider is the uniform distribution on the inputs of Permutated-GIP, i.e. the uniform distribution on all inputs $(z_1, z_2, \ldots, z_p, \pi) \in \{0, 1\}^{dp} \times S_p \times [m]$ where $S_p \times [m]$ is the set of all permutations of $N = pm$ inputs to GIP. We will write $\mu$ as $Z \times U$, where $Z$ is the uniform distribution on $\{0, 1\}^{pm}$ and $U$ is the uniform distribution on $S_p \times [m]$. The random permutation $\pi$ induces a random partition of the bits of $z$ into blocks of size $p$; $U_1, U_2, \ldots, U_m$ where each $U_i$ contains bits $z_{\pi}(1, i), z_{\pi}(2, i), \ldots, z_{\pi}(p, i)$.

We will apply the reduction to communication complexity given by Proposition 3 to obtain our lower bound. To do this we will break the branching program for Permutated-GIP into $r$ layers consisting of many time steps and randomly assign each layer to the party that will simulate the layer. The following lemma will be useful in arguing that in the course of this assignment, it is likely that a block of $p$ tuples in the Permutated-GIP input distribution can be placed on the frontheads of $p$ different players.

**Lemma 7.** For every $\delta > 0$, there is a $p_\delta > 0$ such that for every integer $p \geq p_\delta$ and $d \leq \frac{1}{8} p \log p$, the following holds: Let $G = (L, R)$ be a bipartite graph, where $|L| = |R| = p$. For each $i \in L$, repeat the following process $d_i \leq d$ times: independently at random choose a node $j$ from $R$ with probability $1/p$ and add edge $(i, j)$ if it is not already present. Let $H$ be the graph resulting from subtracting $G$ from $K_{p,p}$. Then $H$ has a perfect matching with probability at least $1 - \delta$. In particular, if $p \geq 69$, this probability is at least 15/16.

Note that Lemma 7 is asymptotically tight with respect to $d$ since, by the standard coupon collector analysis, for any $c > 1/\ln 2$ and $p \geq cp \log p$, the probability that even a single left vertex has a neighbor in $H$ goes to 0 as $p$ goes to infinity. Indeed its proof follows from the fact that below the coupon-collector bound the complement graph is a random bipartite graph of relatively large left degree and hence likely contains a matching.

**Proof:** By Hall’s theorem we can upper bound the probability that there is no perfect matching in $H$ by the probability that there is some $S \subseteq L$ with $1 \leq |S| \leq p - 1$ and $|N(S)| \leq |S|$. (Any witnessing set $S'$ for Hall’s Theorem must be non-empty and the case that $|S'| = p$ is included in the probability that there is a set $S$ with $|N(S)| \leq |S| = p - 1$.) Fix $S \subseteq L$, let $|S| = s$, and fix $T \subseteq R$ such that $|T| = s$. Now $N(S) \subseteq T$ in $H$ iff every $i$ in $S$ has an edge to every $j \in R \setminus T$ in the original graph $G$. $(i, j)$ is not an edge in $G$ if $j$ is not one of the $d_i$ choices for $i$; thus, we have $\Pr[(i, j) \text{ is an edge in } G] = 1 - \left(1 - \frac{1}{p}\right)^{d_i} \leq 1 - \left(1 - \frac{1}{p}\right)^d \leq 1 - 4^{-d/p} \leq 1 - \frac{1}{p^{1/4}}$, since $d_i \leq d \leq \frac{1}{8} p \log p$ and $\left(1 - \frac{1}{p}\right)^d \geq 4^{-d/p}$ for $p \geq 2$.

For each $j \in R \setminus T$, these events are negatively correlated, hence $\Pr[\forall j \in R \setminus T, (i, j) \text{ is an edge in } G] \leq \left(1 - \frac{1}{p^{1/4}}\right)^{p-s}$. Since the choices for each $i \in S$ are independent, it follows that:

$$\Pr[\forall i \in S, \forall j \in R \setminus T, (i, j) \text{ is an edge in } G] \leq \left(1 - \frac{1}{p^{1/4}}\right)^{s(p-s)}.$$

By a union bound, we have

$$\Pr[\exists S \subseteq L, T \subseteq R, \text{ s. t. } |T| = |S| \text{ and } N(S) \subseteq T] \leq \sum_{s=1}^{p-1} \binom{p}{s} \left(1 - \frac{1}{p^{1/4}}\right)^{(p-s)s} \leq \sum_{1 \leq s \leq p/2} \binom{p}{s}^2 \left(1 - \frac{1}{p^{1/4}}\right)^{sp/2} + \sum_{p-1 \geq s \geq p/2} \binom{p}{s}^2 \left(1 - \frac{1}{p^{1/4}}\right)^{(p-s)p/2} \leq 2 \sum_{1 \leq s \leq p/2} \left[p^2(1 - \frac{1}{p^{1/4}})^{p/2}\right]^s \leq 2 \sum_{s \geq 1} \left[p^2 e^{-p^{3/4}/2}\right]^s \leq 2 \frac{p^2 e^{-p^{3/4}/2}}{1 - p^2 e^{-p^{3/4}/2}} \leq \delta$$
provided that \( p \geq p_6 \), where \( p_6 \) is a constant such that 
\[
\frac{2p_6^2 e^{-p_6^2/2}}{1-p_6^2 e^{-p_6^2/2}} \leq \delta.
\]
For \( \delta = \frac{1}{16} \), \( p_6 \leq 69 \). Therefore, \( H \) has a perfect matching with probability at least 15/16, for \( p \geq 69 \).

The following is our main lemma showing that we can convert \( z \)-oblivious branching programs approximating \( \text{Permuted-GIP} \) under the hard distribution \( \mu \) to an efficient communication protocol for \( \text{GIP}_{p,m'} \) under the uniform distribution.

**Lemma 8.** Let \( p, m > 0 \) be positive integers and let \( N = mp \). Assume that \( 69 \leq p \leq \sqrt{N}/2 \) and let \( k \leq \frac{1}{10} p \log p \). Let \( B \) be a \( z \)-oblivious branching program of \( z \)-length \( T \leq kN \) and \( z \)-width \( W \) that approximates \( \text{Permuted-GIP}_{p,m} \) with error at most \( \epsilon \) under the uniform distribution. Then, for \( m' = \frac{p}{\sqrt{p}} \) and \( e' = \epsilon + e\cdot \frac{p}{2p+3} + \frac{1}{p} + \frac{\delta}{8} \), under the uniform distribution on inputs there is a deterministic \( \epsilon' \)-error NOF \( p \)-party protocol for \( \text{GIP}_{p,m'} \) of complexity at most \( k^2 p^{2p+3} \log W \).

**Proof:** Let \( I = [N] \) be the set of input positions in \( z \). Since \( B \) is \( z \)-oblivious we can let \( s \) be the sequence of at most \( kN \) elements from \( I \) that \( B \) queries in order. Divide \( s \) into \( r \) equal segments \( s_1, s_2, \ldots, s_r \), each of length \( \frac{kn}{r} \), where \( r \) will be specified later. Independently assign each segment \( s_i \) to a set in \( A_1, A_2, \ldots, A_p \) with probability \( 1/p \). Denote this probability distribution by \( \mathcal{A} \). Each \( A_j \) represents the elements of \( I \) that \( B \) will have access to and hence we will place a subset of \( I \setminus A_j \) on the forehead of party \( j \). Since the sets \( I \setminus A_j, j = 1, \ldots, p \), might overlap, they might not form a partition of \( I \) into \( p \) sets.

The distribution \( \mu = (Z, \mathcal{U}) \) on \( \text{Permuted-GIP}_{p,m} \) inputs randomly divides the elements of \( I \) into \( m \) blocks \( U_1, U_2, \ldots, U_m \), each of size \( p \). We calculate the probability, under the random assignment of segments to parties and \( z \)-bits to blocks, that we obtain a relatively large number of blocks such that, for every party \( j \), each block contains exactly one \( z \)-bit not belonging to \( A_j \). We bound the probability that a block has \( z \)-bits such that:

1. they do not occur too frequently in the sequence,
2. their assignments to parties are independent, and
3. each \( z \)-bit can be placed on the forehead of a different party.

**Claim 9.** Except with probability at most \( e^{-m/2^{p+3}} \) over \( \mathcal{U} \), all bits in at most \( m/2^{p+2} \) blocks are read at most \( 2k \) times in \( s \). For \( \ell \in [m] \),
\[
\Pr_{\mathcal{U}}[ \text{all } z \text{-bits in } U_{\ell} \text{ appear at most } 2k \text{ times in } s ]
\geq (N/2)^{(p)}/(N)^{(p)} = \frac{N/2}{N} \cdots \frac{(N/2 - (p - 1))}{(N - (p - 1))}
\geq 2^{-p} \prod_{i=0}^{p-1} \left( 1 - \frac{i}{N - i} \right) > \frac{1}{2p} \left( 1 - \frac{2p^2}{N} \right) \geq 1 \frac{1}{2p+1}
\]

since \( p \leq \sqrt{N}/2 \). Hence, the expected number of such blocks is at least \( \frac{m}{2^{p+2}} \). Let \( \mathcal{E}_1 \) be the event that the number of blocks for which all \( z \)-bits appear at most \( 2k \) times in \( s \) is at least \( \frac{m}{2^{p+2}} \), which is at least half the expected number.

For simplicity here we use a single index \( i \) to select the bits of \( z \). Let \( B_i \) be the block that \( z \)-bit \( i \) falls into according to the distribution \( \mathcal{U} \). Let \( Y \) be the number of blocks for which all \( z \)-bits appear at most \( 2k \) times in \( s \). Then \( Y_1 = \mathbb{E}[Y][B_1, B_2, \ldots, B_p] \) is a Doob’s martingale, with \( Y_0 = \mathbb{E}[Y] \geq \frac{m}{2^{p+2}} \) and \( Y_m = Y \). Let \( \mathcal{E}_1 \) be the event that \( Y \) is at least \( m/2^{p+2} \). Then, by the Azuma-Hoeffding inequality, we have
\[
\Pr_{\mathcal{U}}[\mathcal{E}_1] = \Pr_{\mathcal{U}}[|Y_m - Y_0| \geq \frac{m}{2^{p+2}}] \\
\leq e^{-2 \left( \frac{m}{2^{p+2}} \right)^2} = e^{-m/2^{2p+3}}.
\]

**Claim 10.** Except with probability at most \( 1/p \) in \( \mathcal{U} \), there are at most \( m/2^{p+3} \) blocks in which any two \( z \)-bits are queried in the same segment.

Let \( t(i) \) be the number of segments in which \( z \)-bit \( i \) appears, and write \( i \sim i' \) if \( z \)-bits \( i \) and \( i' \) appear in the same segment at least once. Then \( \sum_i t(i) \leq kN \) and the number of \( i' \) such that \( i \sim i' \) is at most \( t(i)kN/r \). A \( z \)-bit \( i \) is in a given block \( U_{\ell} \) with probability \( 1/m \) and the events that \( i \) and \( i' \) are mapped there are negatively correlated. Hence for every \( \ell \in [m] \), we have
\[
\Pr_{\mathcal{U}}[\exists i, i' \in U_{\ell} \text{ such that } i \sim i'] \\
\leq \sum_{i} \sum_{i' \sim i} \frac{1}{m^2} \leq \sum_{i} \frac{t(i)kN}{r} \frac{1}{m^2} = \frac{k^2N^2}{rm^2} = \frac{p^2k^2}{r}.
\]
Setting \( r = 8k^2p^{2p+2} \), the expected number of such bad blocks is at most \( m/(p^{2p+3}) \). Hence, by Markov’s inequality,
\[
\Pr_{\mathcal{U}}[\# \text{ of blocks with } \sim \text{-}z \text{-bits } \geq m/2^{p+3}] \leq 1/p.
\]

Let \( \mathcal{E}_2 \) be the event that there are at least \( m'' = m/2^{p+3} \) blocks such that each \( z \)-bit in these blocks is read at most \( 2k \) times and no two \( z \)-bits in any given block are read in the same segment. Call these blocks good. Then the above claims imply that \( \Pr_{\mathcal{U}}[\mathcal{E}_2] \leq \frac{1}{p} \).
assigning to each class one \( z \)-bit of each block not in \( I' \) and dividing the \( N \) values representing \( \pi \) equally among the parties. Applying Proposition 3 with \( r = k^2 p^{3p+3} \), we obtain a deterministic NOF \( p \)-party protocol of complexity \( k^2 p^{3p+3} \log W \) for \( \text{Permuted-GIP}_{p,m} \) with error \( \epsilon' = \epsilon + \epsilon_1 \) under distribution \( \mu' \).

We reduce the communication problem for \( \text{GIP}_{p,m} \) under the uniform distribution to that for \( 1GA_p \) under \( \mu' \) by observing that the inputs to the \( \text{GIP}_{p,m'} \) problem on a uniformly random input can be embedded into the unfixed blocks of the \( \text{Permuted-GIP}_{p,m} \) instances induced by the permutation \( \pi \) given by the distribution \( \mu' \).

We now apply the following lower bound for \( \text{GIP}_{p,m} \) under the uniform distribution.

**Proposition 11.** [6] \( D_{\text{uniform}}(\text{GIP}_{p,m}) \) is \( \Omega(m/4p + \log(1 - 2\epsilon)) \).

**Theorem 12.** Let \( \epsilon < 1/2 \). There is a \( p \leq \log((m/S)) \) such that if a randomized \( z \)-oblivious branching program computes \( \text{Permuted-GIP}_{p,m} \) with time \( T \), space \( S \) and error at most \( \epsilon \), then \( T = \Omega\left( N \log\left(\frac{N}{s}\right) \log \log \left(\frac{N}{s}\right)\right) \) where \( N = pm \).

**Proof:** Let \( \bar{B} \) be a \( z \)-oblivious randomized branching program computing \( \text{Permuted-GIP}_{p,m} \). By standard probability amplification which increases the width by an additive constant and the time by a constant factor we can assume without loss of generality that the error \( \epsilon \) of \( \bar{B} \) is \( < 1/5 \). Apply Yao’s Lemma to \( \bar{B} \) using distribution \( \mu \) to obtain a deterministic \( z \)-oblivious branching program \( B \) with the same time and space bounds that computes \( \text{Permuted-GIP}_{p,m} \) with error at most \( \epsilon \) under \( \mu \).

Let \( T = kN \) and let \( p \) be the smallest integer \( \geq 69 \) such that \( k \leq \frac{1}{\log p} \log p \). If \( p \geq \log(N/S) / 4 \) then the result is immediate so assume without loss of generality that \( 69 \leq p < \log(N/S) / 4 \). Let \( \epsilon_1 = e^{-\frac{1}{2p^{1/2}}} + \frac{1}{p^{1/2}} \) and which is \( < 1/5 \) for these values of \( p \).

Since \( 69 \leq p \leq \sqrt{N}/2 \) we can combine Lemma 8 and Proposition 11 to say that there is a constant \( C \) independent of \( N \) and \( p \) such that

\[
k^2 p^{3p+3} \log W \geq C\left(\frac{m'}{4p} + \log(1 - 2\epsilon')\right),
\]

where \( m' = m/2p^{4} = \frac{N}{p^{2p^{3}}} \) and \( \epsilon' = \epsilon + \epsilon_1 \leq 2/5 \).

Rewriting \( m' \) and \( k \) in terms of \( N \) and \( p \) and using \( S \geq \log W \), we obtain \( Sp^{2p^{3}} \geq C_{1} n^{4-2p} \), for some constant \( C_{1} \). Simplifying and taking logarithms, we have \( p \geq C_{2} \log\left(\frac{N}{s}\right) \), for some constant \( C_{2} \). Since \( p \) is the smallest integer \( \geq 69 \) such that \( k \leq \frac{1}{\log p} \log p \), we have \( k \geq C_{3} \log\left(\frac{N}{s}\right) \log \log \left(\frac{N}{s}\right) \) for some constant
Let \( w \) be an integer and for \( \ell = w, 2w \), identify the elements of \( \mathbb{F}_{2^\ell} \) with \( \{0,1\}^\ell \). The construction uses the Feistel operator \( F_f \) on \( 2w \) bits which maps \((x,y)\) for \( x,y \in \{0,1\}^w \) to \( (y, f(x) \oplus y) \) where \( f : \{0,1\}^w \rightarrow \{0,1\}^w \). Define a family \( \mathcal{F}_t^w \) of permutations on \( \mathbb{F}_{2^{2w}} \) as the set of all functions constructed as follows: For each independent choice of \( h_1, h_2 \) from \( \mathcal{F}_{2,2^{2w}} \) and \( f_1, f_2 \) from \( \mathcal{F}_{t,2^w} \) define the permutation

\[
\pi_{h_1,h_2,f_1,f_2} = h_2^{-1} \circ F_{f_2} \circ F_{f_1} \circ h_1.
\]

Observe that \( 8w + 2tw \) bits suffice to specify an element of \( \mathcal{F}_t^w \).

**Proposition 14.** ([18] Corollary 8.1) Let \( w \) be an integer and \( t \) be an integer. Then \( \mathcal{F}_t^w \) is \( \delta \)-almost \( t \)-wise independent family of permutations on \( \mathbb{F}_{2^w} \) for \( \delta = t^2/2^w + t^2/2^{2w} \) that also forms a pairwise independent family of permutations.

**Definition 2.** Let \( N \) be a positive integer and \( n = 2^w \) be the largest even power of 2 such that \( n + \log^2 n \leq N \). Let \( p \) be a power of 2 such that \( 2 \leq p \leq \frac{1}{8} \log n \) (of which the are fewer than \( \log \log n \) possibilities).

Define \( \text{GIP-MAP}_N : \{0,1\}^N \rightarrow \{0,1\} \) as follows:
We interpret input bits \( n + 1, \ldots, n + \log \log n \) as encoding the value \( p \leq \frac{1}{8} \log n \) and the next \( 8w + 4pw \leq \frac{3}{4} \log^2 n \) bits as encoding a permutation \( \pi \) from \( \mathcal{F}_t^w \) which we identify with permutation on \( \{n\} \).

\[
\text{GIP-MAP}_N(x_1 x_2 \ldots x_n, p, \pi) = \text{GIP}_{p,n}(z_1, z_2, \ldots, z_p)
\]
where \( z_i = x_{\pi((i-1)n/p+1)} \ldots x_{\pi(in/p)} \) for \( i = 1, \ldots, p \).

**Proposition 15.** \( \text{GIP-MAP}_N \) is computable by a deterministic Boolean branching program using time \( N \) and \( O(\log^2 N) \) space.

**Proof:** The program begins with a full decision tree that first reads the bits of the encoding of \( p \) and then the bits encoding \( \pi \). At each leaf of the tree, the program contains a copy of the width 4 branching program computing \( \text{GIP}_{p,n/p} \) where variable \( z_{ij} \) is replaced by \( x_{\pi((i-1)n+j)} \).

We obtain the following time-space tradeoff lower bound for \( \text{GIP-MAP}_N \).

**Theorem 16.** Let \( \epsilon < 1/2 \). Any randomized oblivious Boolean branching program computing \( \text{GIP-MAP}_N \) with time \( T \), space \( S \) and error at most \( \epsilon \) requires then

\[
T = \Omega \left( N \log \left( \frac{N}{S} \right) \log \log \left( \frac{N}{S} \right) \right).
\]

**Proof:** The proof follows the basic structure of the argument for \( \text{Permuted-GIP}_{p,m} \), except that we now
fix the $p$ that is part of the input. Let $n$ be given as in the definition of $GIP_{MAP_N}$. The $\delta$-almost 2p-wise independence of the permutation ensures that the probability that a block of the permuted $GIP_{p,n/p}$ problem has all its variables accessed at most $2k$ times is roughly $2^{-p}$ and that these events are roughly pairwise independent. The pairwise independence of the permutation ensures that two variables in a block are unlikely to be assigned to the same segment.

Let $B$ be a randomized oblivious branching program computing $GIP_{MAP_N}$ and assume wlog that the error $\epsilon$ of $B$ is $< 1/5$. We can assume wlog that $\log(n/S) \geq 2^{10}$ or the result follows immediately. Otherwise let $p$ be the largest power of 2 such that $p \leq \frac{1}{5} \log(n/S)$. Then $p \geq 69$. Let $\mu_p$ be the uniform distribution over the input bits to $GIP_{MAP_N}$ conditioned on the fixed value of $p$. Apply Yao’s Lemma to $B$ using distribution $\mu_p$ to obtain a deterministic oblivious branching program $B$ with the same time and space that computes $GIP_{MAP_N}$ with error at most $\epsilon$ under $\mu_p$.

Suppose that the time of $B$, $T \leq kn$ where $k = \frac{1}{\log p}$. Since $B$ is oblivious we can let $s$ be the sequence of at most $kn$ elements from the set of input positions $I = [n]$ that $B$ queries, in order. (We do not include the input positions queried outside of $[n]$ since their values will eventually be fixed.) Divide $s$ into $r$ equal segments $s_1, s_2, \ldots, s_r$, each of length at most $\frac{kn}{r}$, where $r$ will be specified later. Independently assign each segment $s_i$ to a set in $A_1, A_2, \ldots, A_p$ with probability $1/p$. Denote this probability distribution $A$. Each $A_j$ represents the elements of $L$ that party $j$ will have access to and hence we will place a subset of $I \setminus A_j$ on the forehead of party $j$. Since the sets $L \setminus A_j$, $j = 1, \ldots, p$, might overlap, they might not form a partition of $I$ into $p$ sets.

The permutation $\pi$ randomly divides $[n]$ into $m = n/p$ blocks $U_1, U_2, \ldots, U_m$, each of size $p$ where block $U_j$ contains the $j$th bits of the vectors $z_1, z_2, \ldots, z_p$ as given in the definition of $GIP_{MAP_N}$. By construction, the distribution of $\pi$ is $\delta$-almost 2p-wise independent for $\delta = 8p^2/\sqrt{n}$.

We now follow many of the lines of the remainder of the proof of Lemma 8 and give full details where the proofs differ. The key difference in the calculations is that we no longer have a truly random permutation. The parameter $n$ here corresponds to $N$ in the proof of Lemma 8.

As before, we calculate the probability, under the random assignment of segments to parties and elements of $[n]$ to blocks, that we obtain a relatively large number of blocks such that, for every party $j$, each block contains exactly one element of $[n]$ not belonging to $A_j$. To do this, we bound the probability that a block has elements of $[n]$ such that:

1. they do not occur too frequently in the sequence,
2. their assignments to parties are independent, and
3. each element can be placed on the forehead of a different party.

In the proof of Lemma 8, conditioned on (1) and (2), the argument for (3) is independent of the choice of $\pi$ and depends only on the randomness of the assignment of segments to parties. The proof of (2) depends only on the pairwise independence of $\pi$ which is guaranteed here by Proposition 14. Only the proof of part (1) needs to be modified substantially.

As before, we first remove all input indices that appear more than $2k$ times in the sequence $s$. By Markov’s inequality, at least half the input indices appear at most $2k$ times in $s$. Let the first $n/2$ elements of this set be $G$.

Therefore, for $\ell \in [m]$, let $Y_\ell$ be the indicator function for the event that $U_\ell \subseteq G$. Then since $\pi$ is $\delta$-almost 2p-wise independent

$$\Pr_{\mu_p}[Y_\ell = 1] = \Pr_{\mu_p}[U_\ell \subseteq G] \geq \frac{(n/2)(p)/n^2 - \delta}{2^{-p} - 2^{-p-1}p^2/n - \delta}$$

where $\delta' = \delta + 2^{-p-1}p^2/n < 9p^2/\sqrt{n}$. Similarly and more simply, $\Pr_{\mu_p}[Y_\ell = 1] \leq 2^{-p} + \delta \leq 2^{-p} + \delta'$. Let $E_1$ be the event that the number of blocks for which all elements appear at most $2k$ times in $s$ is at least $m'' = \frac{n}{p2^{-p}+\delta'}$.

We use the second moment method to upper bound $\Pr_{\mu_p}[E_1]$. Let $Y$ be the number of blocks for which all elements appear in $G$. Then $Y = \sum_{\ell \in [n/p]} Y_\ell$ and $\mathbb{E}(Y) = \mathbb{E}(Y) + \sum_{\ell \neq \ell'} \text{Cov}(Y_\ell, Y_{\ell'})$ where $\text{Cov}(Y_\ell, Y_{\ell'}) = \Pr[Y_\ell Y_{\ell'} = 1] - \Pr[Y_\ell = 1] \Pr[Y_{\ell'} = 1]$. Since the outputs of $\pi$ are $\delta$-almost 2p-wise independent, we have: $\Pr[Y_\ell Y_{\ell'} = 1] = \Pr[U_\ell \cup U_{\ell'} \subseteq G] \leq 2^{-2p} + \delta \leq 2^{-2p} + \delta'$. Therefore

$$\text{Var}(Y) \leq \frac{n}{p} 2^{-p} + \frac{n}{p} \delta'$$

$$= \frac{n}{p} 2^{-p} + \frac{n}{p} \delta' + \frac{n}{p} (2^{-p} - \delta')$$

$$\leq \frac{n}{p} 2^{-p} + \frac{2n^2}{p^2} \delta'.$$
Now \( E_1 \) holds if \( Y \geq m'' = \frac{n}{2^{p+\epsilon}} \geq \mathbb{E}(Y) - \frac{n}{p}(2^{-p-1} + \delta') \). So by Chebyshev’s inequality, we have

\[
\Pr_{\mu}[E_1] \leq \Pr \left[ \left| Y - \mathbb{E}(Y) \right| \geq \frac{n}{p}(2^{-p-1} + \delta') \right] \\
\leq \frac{\text{Var}(Y)}{\left( \frac{n}{p}(2^{-p-1} + \delta') \right)^2} \\
\leq \frac{(n/p)^2 2^n + 2(n/p)^2 \delta'}{(n/p)^2 (2^{-p-1} + \delta')^2} \\
\leq \frac{p2^{p+2}}{n} + 2\delta' \\
= \frac{p2^{p+2} + 18p^2}{n}.
\]

Since \( 69 \leq p \leq \frac{1}{8} \log n \), we obtain \( \Pr_{\mu}[E_1] \leq 2^{-3p} \).

As in the proof of Lemma 8 let \( t(i) \) be the number of segments in which \( i \) appears, and write \( i \sim i' \) if elements \( i \) and \( i' \) appear in the same segment at least once. Then the number of \( i' \) such that \( i \sim i' \) is at most \( t(i)kn/r \) and \( \sum_i t(i) = kn. \) By construction the random permutation \( \pi \) is pairwise independent and hence it maps any two input bits \( i \neq i' \in [n] \) to two randomly chosen distinct points in \([n]\). Therefore the probability that they are both chosen for some block \( U_j \) is precisely \( p(p - 1)/n(n - 1) \leq p^2/n^2. \) Hence for every \( \epsilon \in [n/p] \), we have

\[
\Pr_{\mu}[\exists i, i' \in U_{\epsilon} \text{ such that } i \sim i'] \\
\leq \sum_i \sum_{i' \sim i} \frac{p^2}{n^2} = \sum_i \frac{t(i)kn}{r} \frac{p^2}{n^2} = \frac{k^2p^2n^2}{rnn^2} = \frac{k^2p^2}{r}.
\]

Setting \( r = k2^p3^{2p+2} \), the expected number of such blocks is at most \( \frac{2^{3p}}{r2^{p+\epsilon}} = m''/(2p) \). Hence, by Markov’s inequality, \( \Pr_{\mu}[\text{the number of blocks with } \sim \text{ tuples } \geq m''/2] \leq \frac{1}{p}. \) By \( E_2 \) be the event that there are at least \( \frac{m''}{2} = \frac{n}{p2^{p+\epsilon}} \) blocks such that each bit in these blocks is read at most \( 2k \) times and no two bits in any block are read in the same segment. Then \( \Pr_{\mu}[E_2] \leq 2^{-3p} + 1/p. \)

Conditioned on the event \( E_2 \), the probability that the number of blocks among the \( m' = m''/2 \) blocks guaranteed by \( E_2 \) for which elements that can be placed on the forehead of the \( p \) different parties is independent of the choice of \( \pi \) and depends only on the assignment \( A. \) By the same calculation as that of Lemma 8 with the value of \( m'' \) here, except with a probability of \( 1/8, \) conditioned on \( E_2 \), there are at least \( m' = \frac{n}{8p2^p} \) blocks whose elements can be placed on the foreheads of different parties. Let \( E_3 \) be the probability over the joint distribution of \( \mu \) and \( A \) that there are at least \( m' \) such blocks. The \( \Pr[E_3] \) is at most \( \epsilon_1 = 2^{-3p} + 1/p + 1/8. \)

There must be some choice of \( A = (A_1, \ldots, A_p) \) for which the probability, over the distribution \( \mu_\pi \) on the grouping of blocks, that \( E_3 \) does not occur is at most the average \( \epsilon_1. \) We fix such an \( A. \)

Since the branching program \( B \) correctly computes \( GIP-\text{MAP}_N \) under distribution \( \mu_\pi \) with probability at least \( 1 - \epsilon, \) there must some choice \( \pi \) of the permutation that groups the elements of \([n]\) into blocks with \((\pi, A) \in E_3 \) such that \( B \) correctly computes \( GIP-\text{MAP}_N \) with probability at least \( 1 - \epsilon - \epsilon_1 \) under the distribution \( \mu_\pi \) conditioned on the choice of \( \pi. \) (This conditional distribution is now determined entirely by the uniform distribution over \( \{0, 1\}^n. \))

Let \( I' \), with \(|I'| = m' \) be the set of blocks witnessing event \( E_3. \) By averaging there must be some assignment \( \zeta \) to the blocks not in \( I' \) so that \( B \) correctly computes \( GIP-\text{MAP}_N \) with probability at least \( 1 - \epsilon - \epsilon_1 \) under distribution \( \mu \) conditioned on the choice of the permutation \( \pi \) and assignment \( \zeta. \) Let \( \mu' \) be this conditional distribution which is uniform on the inputs appearing in the blocks of \( I'. \) As in the proof of Lemma 8 we can use the branching program \( B \) to obtain a deterministic \( p\)-party communication protocol of complexity at most \( rS = k^2p^32p^2 + S \) that computes \( \text{GIP}_{p,m} \) with the standard input partition for a uniformly random input in \( \{0, 1\}^{pm} \) with error at most \( \epsilon' = \epsilon + \epsilon_1 < 2/5. \)

Hence, by Proposition 11, there is an absolute constant \( C \) such that \( k^2p^32p^2 + S \geq \frac{Cm'}{4p} = \frac{Cn}{8p2^p}. \) Since \( k = \frac{1}{16}p \log p, \) we obtain \( 2^p2^p3^p \log 2^p \geq 8Cn/S \) which contradicts the assumption that \( p \) is the largest power of \( 2 \) smaller than \( \frac{1}{4} \log(n/S) \) for \( n/S \) sufficiently large.

Our only hypothesis was that \( T \leq kn \) so we must have \( T > kn = \frac{1}{16}np \log p \) which is at least \( cn \log(n/S) \log(n/S) \) for some constant \( c > 0. \) Since \( n = \Theta(N) \), the theorem follows.

6. DISCUSSION

Our results apply to randomized oblivious algorithms and are the largest explicit time-space tradeoff lower bounds known for randomized non-uniform branching programs. However, it would be interesting to extend these bounds to more powerful classes of randomized branching programs, in particular oblivious randomized ones where the probability distribution on the input sequence is independent of the input. We conjecture that \( 1GAP_n \) is also hard for this stronger oblivious randomized model. It is important to note that if we applied Yao’s Lemma directly on this model then we would lose the requirement of obliviousness when the randomness is fixed.
ACKNOWLEDGEMENTS

We are very grateful to Russell Impagliazzo for helping us to clarify the difference between randomized oblivious and oblivious randomized branching programs and for suggesting the approach for separations for Boolean branching programs based on pseudorandom permutations of the generalized inner product.

REFERENCES
