Electronic Colloquium on Computational Complexity, Report No. 104 (2010)

Making RAMs Oblivious Requires Superlogarithmic Overhead

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

Widad Machmouchi*
Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
widad@cs.washington.edu

June 21, 2010

Abstract

We prove a time-space tradeoff lower bound of $T = \Omega\left(n \log(\frac{n}{S}) \log\log(\frac{n}{S})\right)$ for randomized oblivious branching programs to compute $1GAP$, 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.

Ajtai [3, 4] and Damgård, Meldgaard, and Nielsen [11] recently derived simulations of general RAMs by randomized oblivious RAMs with only a polylogarithmic factor increase in time and space. Our lower bound implies that a superlogarithmic factor increase is indeed necessary in any such simulation.

1 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 to model modern computers more closely, are not.

An important reason to be interested in oblivious computation is a concern about *side channel attacks* on cryptographic protocols. In these attacks an adversary can do more than simply eavesdrop on a channel; an adversary can monitor such things as power consumption, the volume of data being transmitted between parts of a chip, or even the operation codes or data addresses being referenced. For an oblivious algorithm, such attacks would yield no information about the input, which can be taken to include any secret key or other sensitive information. This has led to the goal of efficiently converting general algorithms to oblivious ones.

Until recently the only efficient methods to convert general RAM algorithms to oblivious ones required cryptographic assumptions such as access to truly random functions [12, 14, 13]. In recent papers, Ajtai [3, 4] and Damgård, Meldgaard, and Nielsen [11] eliminated the cryptographic assumptions and showed how to do this simulation using only random bits and achieve only a polylogarithmic factor overhead in both time and space. The resulting algorithms are randomized oblivious algorithms in that at each step there

*Research supported by NSF grant CCF-0830626

ISSN 1433-8092

is a probability distribution on the set of operations and operands which is independent of the input. The resulting algorithm in [4] uses $\log^{\gamma}$-wise independent random bits and succeeds with high probability [3, 4]. The resulting algorithm sketched in [11] assumes a random shuffle operation with $O(\log^2 n)$ overhead and always succeeds. This gives rise to a natural question: Is the polylogarithmic increase as in these simulations necessary?

Our main result implies that an increase of this kind is indeed necessary. We show that a very simple problem for deterministic RAM algorithms, the pointer jumping or $1GAP$ problem of out-degree 1 graph reachability, when solved on a randomized oblivious algorithm with sufficiently small constant error, requires time T and S such that $T = \Omega\left(n \log(\frac{n}{S}) \log\log(\frac{n}{S})\right)$. This implies that for any $\delta > 0$ any randomized oblivious algorithm using space $n^{1-\delta}$ requires time $\Omega(n \log n \log\log n)$ which contrasts with the simple space $O(\log n)$ and time n deterministic RAM algorithm and hence only a polylogarithmic overhead in space requires an $\Omega(\log n \log\log n)$ factor overhead in time.

Our lower bound applies 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 1.1. *Let $\epsilon < 1/2$. Randomized oblivious branching programs computing $1GAP_n$ using time T, space S and with error at most ϵ require $T = \Omega\left(n \log(\frac{n}{S}) \log\log(\frac{n}{S})\right)$.*

Since $1GAP_n$ 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 1.2. *Any method for converting 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.*

Deterministic oblivious branching programs have been studied in many contexts. Indeed the much-studied *ordered binary decision diagrams* (or OBDDs) [10] 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 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 for a different Boolean function to $T = \Omega(n \log^2(n/S))$ by Babai, Nisan, and Szegedy [6].

Beame and Vee [9] used a slightly simpler argument related to that in [6] 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 *1GAP* problem. However, that separation does not apply to the randomized setting of the simulations that we are interested in here.

Though a number of time-space tradeoff lower bounds have been proven for natural problems in NP for general deterministic and nondeterministic [8, 1, 2] and randomized computation [7], 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 $k = O(\log n)$ [15]. No prior separations of randomized oblivious computations from general computation have been known.

Our argument builds on the ideas of [9] which uses the connection between oblivious branching programs and communication complexity that is implicit in [5] and first made explicit in the context of multiparty communication complexity in [6]. As in [9] we also make use of the fact that inputs to the $1GAP$ problem can encode functions computable by small branching programs, and in particular, the generalized inner product. The key to our argument is a way of extending the worst-case reduction in [9] to a more involved analysis that yields a reduction that works for distributional complexity.

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 the *0-sink node*, a set of n inputs and one output. For the purpose of this paper, we consider single-output branching programs. 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 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. (We write $\log x$ to denote $\log_2 x$.)

A branching program B is said to *compute* 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 μ with error at most ϵ* iff $B(x) = f(x)$ for all but an ϵ-measure of $x \in D^n$ under distribution μ. (For a probability distribution μ we write $supp(\mu)$ denote its support.)

A branching program is *leveled* if the 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 $\log W$. An *oblivious* branching program is a leveled branching program in which the same input symbol is queried by all the nodes at the same level. A *randomized* oblivious branching program $\mathcal{B}$ is a probability distribution over deterministic oblivious branching programs with the same input set. $\mathcal{B}$ computes a function f with error at most ϵ if for every input $x \in D^n$, $\Pr_{B \sim \mathcal{B}}[B(x) = f(x)] \geq 1 - \epsilon$.

Multiparty communication complexity Let $f : D^n \to \{0, 1\}$. Assume 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, hence the computing model is known as the number-on-forehead model.) 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) multiparty communication complexity* of f with respect to the partition $\mathcal{P}$, denoted $C_{\mathcal{P}}(f)$, 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 μ be a probability distribution over D^n. The *(μ, ϵ)-distributional communication complexity* of f

with respect to $\mathcal{P}$, denoted $D^{\mu}_{\epsilon,\mathcal{P}}(f)$, 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 μ. Again, we replace the $\mathcal{P}$ by p when $\mathcal{P}$ is a p-partition that is understood in the context.

Pointer Jumping and Generalized Inner Product We consider the out-degree 1 directed graph reachability problem, $1GAP$, which is also known as the pointer jumping problem. Define $1GAP_n : [n+1]^n \rightarrow \{0,1\}$ such that $1GAP_n(x) = 1$ iff there is a sequence of indices $i_1, 1_2, \ldots, i_\ell$ such that $i_1 = 1$, $i_\ell = n+1$ and $x_{i_j} = i_{j+1}$ for all $j = 2, \ldots \ell - 1$. The $1GAP_n$ 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 $1GAP_n$ for randomized oblivious branching programs to that of the *generalized inner product* problem $GIP_{p,n}$ for a suitable value of p. $GIP_{p,n} : (\{0,1\}^n)^p \rightarrow \{0,1\}$ is given by $GIP_{p,n}(z_1, \ldots, z_p) = \oplus_{j=1}^{n} \bigwedge_{i=1}^{p} z_{ij}$. The standard input partition for $GIP_{p,n}$ 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 (read-once) branching program.

3 Time-Space Tradeoff Lower Bound for $1GAP_n$

Proof Overview In order to derive lower bounds for computing $1GAP_n$ on randomized oblivious branching programs we use the following easy half of Yao's minimax lemma relating randomized and distributional complexity.

Proposition 3.1. *Let $\widetilde{B}$ be a randomized oblivious branching program computing a function f using time T and space S, with error at most ϵ. Then, for every distribution μ on the inputs, there exists a deterministic oblivious branching program B using time at most T and space at most S that approximates f with error at most ϵ, where the inputs are weighed by μ.*

Therefore, it is enough to find a distribution μ such that every deterministic oblivious branching program approximating $1GAP_n$ under μ will have a large time-space tradeoff. We describe this distribution μ in detail in subsection 3.1. The distribution will arrange that the $1GAP_n$ graph to simulate the width 4 branching program computing the generalized inner product, $GIP_{p,n}$, applied to uniformly random inputs and it will do so in a way that the levels of the branching program correspond to random tuples of nodes in the $1GAP_n$ graph.

Under this distribution on the inputs, we consider a deterministic oblivious branching program approximating $1GAP_n$, and we transform it into a p-party number-on-the-forehead communication protocol by fixing both the assignment of levels of the $GIP_{p,n}$ branching program to tuples of nodes and by fixing the values of some of the blocks of p levels. This protocol also approximates $1GAP_n$ with slightly larger error ϵ' under a new distribution μ' that fixes these aspects of μ. As explained in subsection 3.2, this new protocol communicates a small number of bits that depends on the time and space used by the oblivious branching program approximating $1GAP_n$.

Finally, in subsection 3.3, we show the reduction of $GIP_{p,n}$ to $1GAP_n$ so that a communication protocol approximating $1GAP_n$ can be transformed into a protocol approximating $GIP_{p,n}$ and use the distribution lower bound for $GIP_{p,n}$ given in [6] to derive the time-space tradeoff lower bound.

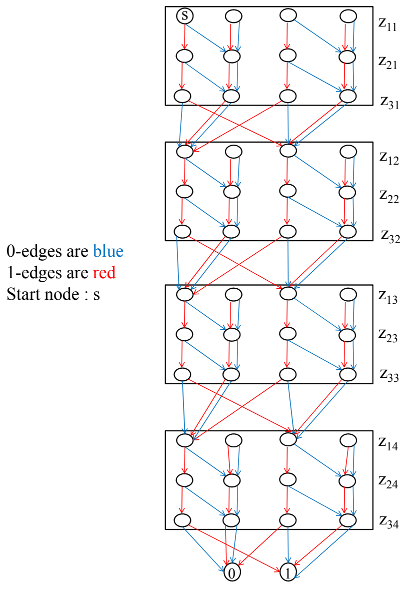

Figure 1: A width 4 branching program computing $GIP_{3,4}$.

3.1 Hard distribution on the inputs

To prove lower bound on the complexity of randomized oblivious branching programs computing $1GAP_n$ with high probability, we give a distribution μ on the inputs such that any deterministic oblivious branching program B such that $\mathbb{E}_{x\sim\mu}\mathbf{1}_{\{B(x)\neq 1GAP(x)\}} \leq \epsilon$, requires a large time-space tradeoff.

As described above, the distribution on $1GAP_n$ graphs will simulate a width 4 branching program computing the generalized inner product, $GIP_{p,n}$, an example of which is given in Figure 1, applied to uniformly random inputs. The distribution will do so in a way that the levels of the branching program correspond to random tuples of nodes in the $1GAP_n$ graph.

Let $p > 0$ be a positive integer. Assume for simplicity and without loss of generality that $4p$ divides n. Divide $\{1, \ldots, n\}$ into $N = \frac{n}{4}$ tuples of 4 consecutive nodes each: $(1, 2, 3, 4), \ldots, (n-3, n-2, n-1, n)$. The path starting at node 1 will reach precisely one node in each tuple. Randomly group the tuples into blocks, each of size p. Hence, there are $M = \frac{n}{4p}$ blocks $U_1, U_2, \ldots, U_M$, each containing p tuples. Within each block, order the tuples by increasing node numbers. With each choice of grouping we associate some fixed order to the blocks such that $(1, 2, 3, 4)$ is (the first tuple) in U_1 and $(n-3, n-2, n-1, n)$ is (the last tuple) in U_M. We denote the distribution on the groupings by $\mathcal{U}$.

Let $z_1, z_2, \ldots, z_p$ be chosen uniformly at random from $\{0,1\}^M$; i.e., $z_i \in_U \{0,1\}^M$, for $i = 1, \ldots, p$. We denote this distribution as $\mathcal{Z}$. We connect the nodes in the graph according to the values of the z_i's as in Figure 2, which is described more formally as follows:

- There are no edges between nodes in the same tuple.
- Within each block U_j, for $i = 1, \ldots, p-1$, we connect tuple $i = (a, a+1, a+2, a+3)$ to tuple $i+1 = (b, b+1, b+2, b+3)$ according to the value of z_{ij}: If $z_{ij} = 1$, add directed edges (a, b), $(a+1, b+1)$,$(a+2, b+2)$ an $(a+3, b+3)$. If $z_{ij} = 0$, add directed edges $(a, b+1)$, $(a+1, b+1)$,$(a+2, b+3)$ an $(a+3, b+3)$.

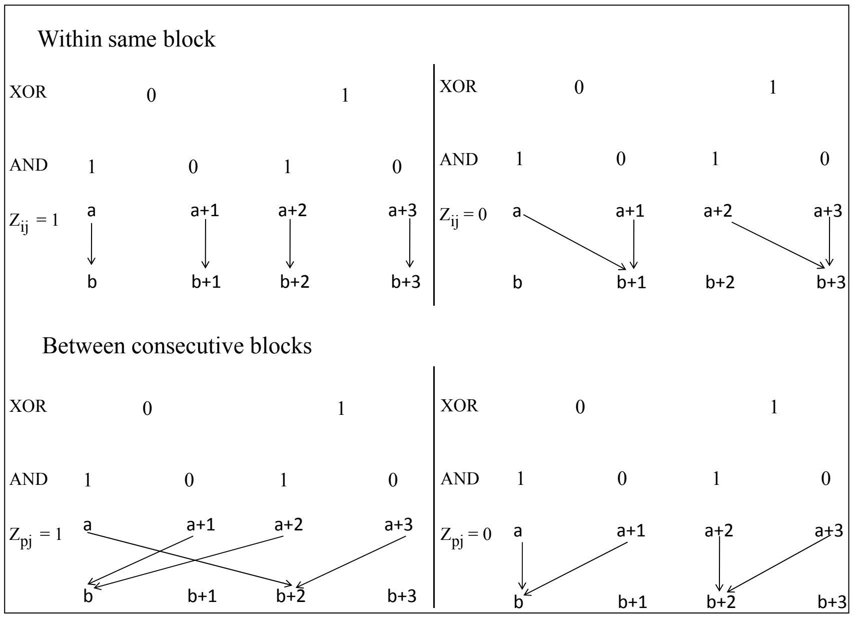

Figure 2: Edges between the nodes according to the value of the $z_i's$.

- We connect consecutive blocks U_j and U_{j+1} according to the value of z_{pj}. Let $(a, a+1, a+2, a+3)$ be the p^{th} tuple in U_j and $(b, b+1, b+2, b+3)$ the first tuple in U_{j+1}. If $z_{pj} = 1$, we add $(a, b+2)$, $(a+1, b)$, $(a+2, b)$ and $(a+3, b+2)$. If $z_{pj} = 0$, we add (a, b), $(a+1, b)$, $(a+2, b+2)$ and $(a+3, b+2)$.
- The p^{th} tuple of U_M is $(n-3, n-2, n-1, n)$. We add the directed edges $(n-2, n-2)$ and $(n, n+1)$, independent of the value of z_{pM}. If $z_{pM} = 1$, we add the directed edges $(n-3, n+1)$ and $(n-1, n-1)$. If $z_{pM} = 0$, we add the directed edges $(n-3, n-3)$ and $(n-1, n+1)$.

Therefore, the probability distribution μ on inputs to $1GAP_n$ can be defined as a deterministic function of two independent probability distributions $\mathcal{U}$ and $\mathcal{Z}$, where $\mathcal{U}$ is given by the random grouping of tuples into blocks and $\mathcal{Z}$ is given by the random choice of $z_1, z_2, \ldots, z_p$.

The construction above emulates computing $GIP_{p,M}$ on the input $(z_1, z_2, \ldots, z_p)$. The first two nodes in each tuple represent the value of the AND up to the corresponding z_{ij} given that the current value of the XOR is 0. The second two nodes represent the value of the AND up to the corresponding z_{ij} given that the current value of the XOR is 1. The tuples within each block are connected in a way to compute the AND of the variables corresponding to those tuples. The connections between blocks are established in a way to compute the current value of the XOR.

3.2 Branching Program Complexity and NOF Multiparty communication

Let B be a deterministic oblivious branching program of length kn and width W computing a function from D^n to $\{0,1\}$. Since B is oblivious, it can be viewed as a sequence π of queries to input symbols of length kn.

DEFINITION 3.1. *[9] Let s be a sequence of values from a domain N and let $\mathcal{P}'$ be a partition of a subset $N' \subseteq N$. The number of alternations of s with respect to $\mathcal{P}'$ is the minimum r such that $s = s_1 s_2 s_3 \ldots s_r$ and each s_i does not contain any elements from at least one class in $\mathcal{P}'$. We say that a partition $\mathcal{P}$ extends $\mathcal{P}'$ iff each class in $\mathcal{P}$ contains a unique class of $\mathcal{P}'$.*

The following proposition is adapted from [6, 9] for oblivious BPs approximating a function.

Proposition 3.2. *Let $f : D^n \to \{0,1\}$, let $\mathcal{P}'$ be a partition of a subset I' of $[n]$, and let μ be a distribution on D^n that has fixed values for all input variables indexed by $[n] - I'$. Let B be a deterministic oblivious branching program of width W whose query sequence s has at most r alternations with respect to $\mathcal{P}'$ such that B approximates f under a probability distribution μ with error at most ϵ. Then for any partition $\mathcal{P}$ that extends $\mathcal{P}'$ to all of $[n]$, $(r-1)\log(W) + 1 \geq D^{\mu}_{\epsilon,\mathcal{P}}(f)$.*

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. Let s be the query sequence of B and let $s_1, s_2, \ldots, s_r$ be the alternations of s with respect to $\mathcal{P}$. Hence, for every i, s_i does not contain any element of a class P_{j_i} of $\mathcal{P}$, except those in $[n] - I'$. The communication protocol starts at the source node of B and for every $i \in \{1, \ldots, r\}$, party j_i queries the input variables in s_i and then records on the board the last node reached at the end of s_i. These queries are possible since s_i does not contain any of the elements of P_{j_i} except those known to all parties and hence all variables queried in s_i are either known or on the foreheads of parties other than party j_i. To record the last node reached, each player needs $\log W$ bits since B has width W, except for the last player, who needs only one bit to record the output of B. $\square$

In our argument we will break the branching program for $1GAP_n$ into layers consisting of many time steps and randomly assign layers to a p-partition of the input variables corresponding 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 $1GAP_n$ input distribution can be placed on the foreheads of p different players.

Lemma 3.3. *For every $\delta > 0$, there is a $p_\delta > 0$ such that for every positive 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 $15/16$.*

Proof. We use Hall's theorem to calculate the probability that G has a perfect matching:

Proposition 3.4. ***(Hall's Theorem)*** *Let $H = (L, R)$ be a bipartite graph. If for every $S \subseteq L$, $|N(S)| \geq |S|$, then H has a perfect matching, where $N(S)$ denotes the neighborhood of S in R.*

Therefore we can upper bound the proability 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 $\Pr[(i,j)$ 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 $\Pr[\forall j \in R \setminus T, (i,j)$ 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

$$\begin{aligned}
&\Pr[\exists S \subseteq L,\ T \subseteq R, \text{ such that } |T| = |S| \text{ and } N(S) \subseteq T] \\
&\leq \sum_{s=1}^{p-1} \binom{p}{s}^2 (1 - \frac{1}{p^{1/4}})^{s(p-s)} \\
&\leq \sum_{1 \leq s \leq p/2} \binom{p}{s}^2 (1 - \frac{1}{p^{1/4}})^{sp/2} + \sum_{p-1 \geq s \geq p/2} \binom{p}{s}^2 (1 - \frac{1}{p^{1/4}})^{(p-s)p/2} \\
&= \sum_{1 \leq s \leq p/2} \binom{p}{s}^2 (1 - \frac{1}{p^{1/4}})^{sp/2} + \sum_{1 \leq p-s \leq p/2} \binom{p}{p-s}^2 (1 - \frac{1}{p^{1/4}})^{(p-s)p/2} \\
&\leq 2 \sum_{1 \leq s \leq p/2} \left[p^2 (1 - \frac{1}{p^{1/4}})^{p/2}\right]^s \\
&\leq 2 \sum_{s \geq 1} \left[p^2 e^{-p^{3/4}/2}\right]^s \\
&\leq \frac{2p^2 e^{-p^{3/4}/2}}{1 - p^2 e^{-p^{3/4}/2}} \\
&\leq \delta
\end{aligned}$$

provided that $p \geq p_\delta$, where p_δ is a constant such that $\frac{2p_\delta^2 e^{-p_\delta^{3/4}/2}}{1 - p_\delta^2 e^{-p_\delta^{3/4}/2}} \leq \delta$. For $\delta = \frac{1}{16}$, $p_\delta \leq 69$. Therefore, H has a perfect matching with probability at least $15/16$, for $p \geq 69$. □

Note that the Lemma 3.3 is asymptotically tight with respect to d since, by the standard coupon collector analysis, for any $c > 1/ln2$ and $p \geq cplogp$, the probability that even a single left vertex has a neighbor in H goes to 0 as p goes to infinity.

Now, we consider the $1GAP_n$ problem and the "hard" distribution μ defined on its inputs given in subsection 3.1. The following lemma shows that for every oblivious branching program approximating $1GAP_n$ under μ, there is p-party NOF communication protocol that approximates $1GAP$ under a distribution μ'.

Lemma 3.5. *Let $1GAP_n : [n+1]^n \to \{0,1\}$ and let B be an oblivious branching program of length $T = kn$ and width W, that approximates $1GAP$ with error at most ϵ under the probability distribution μ. Let p be an integer such that $69 \leq p \leq \sqrt{n}/2$ and $k \leq \frac{1}{16} p \log p$. Then there is a fixed grouping U of tuples into blocks, a p-partition $\mathcal{P}$ with the four nodes from one tuple from each block in each of its classes, a fixed assignment ζ to the $\mathcal{Z}$ values of all but $m = \frac{n}{64p2^p}$ of these blocks, and a distribution μ' resulting from restricting μ to these two assignments such that $32k^2p^3 \log W \geq D^{\mu'}_{\epsilon',\mathcal{P}}(1GAP_n)$, where $\epsilon' = \epsilon + e^{-\frac{n}{32p^2 4^p}} + \frac{1}{p} + \frac{1}{8}$.*

Proof. Divide $\{1,\dots,n\}$ into $\frac{n}{4}$ tuples of 4 consecutive nodes each: $(1,2,3,4),\dots,(n-3,n-2,n-1,n)$. We assume that each node in B reading an input variable ℓ also can read all the input variables that appear in the same tuple as ℓ. Hence, we can assume without loss of generality that B is an $(n+1)^4$-way branching program of length kn and width W that reads entire tuples. Since B is oblivious we can let s be the sequence of kn elements in $L=\{(1,2,3,4),\dots,(n-3,n-2,n-1,n)\}$ that B queries in order. Divide s into r equal segments $s_1,s_2,\dots,s_r$, each of length $\frac{kn}{r}$, where r will be specified later. Independently assign each segment s_i to a set in $A_1,A_2,\dots,A_p$ with probability $1/p$. Denote this probability distribution $\mathcal{A}$. Each A_j represents the elements of L that party j will have access to and hence we will place a subset of $L\setminus A_j$ on the forehead of party j. Since the sets $L\setminus A_j$, $j=1,\dots,p$, might overlap, they might not form a partition of L into p sets.
The distribution $\mu=(\mathcal{U},\mathcal{Z})$ on the $1GAP_n$ inputs randomly divides the tuples into M blocks $U_1,U_2,\dots,U_M$, each of size p. We calculate the probability, under the random assignment of segments to parties and tuples to blocks, that we obtain a relatively large number of blocks such that, for every party j, each block contains exactly one tuple not belonging to A_j. We bound the probability that a block has tuples such that:

- they do not occur too frequently in the sequence.
- their assignments to parties are independent.
- each tuple can be placed on the forehead of a different party.

We first remove all tuples that appear more than $2k$ times in the sequence s. By a simple application of Markov's inequality, at least half the tuples appear at most $2k$ times in s.

For $\ell=1,\dots,M$,

$$\begin{aligned}\Pr_{\mathcal{U}}[\text{ all tuples in } U_\ell \text{ appear at most } 2k \text{ times in } s] &\geq \frac{\binom{n/8}{p}}{\binom{n/4}{p}}\\ &= \frac{n/8}{n/4}\cdots\frac{(n/8-(p-1))}{(n/4-(p-1))}\\ &= \prod_{i=0}^{p-1}\left(\frac{1}{2}-\frac{i}{n/4-i}\right)\\ &\geq \frac{1}{2^p}\left(1-\frac{2p^2}{n}\right)\\ &\geq \frac{1}{2^{p+1}}\end{aligned}$$

since $p\leq\sqrt{n}/2$. Hence, the expected number of such blocks is at least $\frac{n}{4p2^{p+1}}$. Let $\mathcal{E}_1$ be the event that the number of blocks for which all tuples appear at most $2k$ times in s is at least $N=\frac{n}{4p2^{p+2}}$, which is at least half the expected number.

Let B_i be the block that tuple i falls into according to the distribution $\mathcal{U}$. Let Y be the number of blocks for which all tuples appear at most $2k$ times in s. Then $Y_t=\mathbb{E}[Y|B_1,B_2,\dots,B_t]$ is a Doob's martingale, with $Y_0=\mathbb{E}[Y]\geq\frac{n}{4p2^{p+1}}$ and $Y_{\frac{n}{4}}=Y$. Let $\mathcal{E}_1$ be the event that Y is at least N. Then, by the Azuma-Hoeffding inequality, we have

$$\Pr_{\mathcal{U}}[\overline{\mathcal{E}_1}]=\Pr\left[|Y_{\frac{n}{4}}-Y_0|\geq\frac{n}{4p2^{p+2}}\right]\leq e^{-2\frac{\left(\frac{n}{4p2^{p+2}}\right)^2}{n/4}}=e^{-\frac{n}{32p^2 4^p}}.$$

We will only consider blocks such that no two tuples in that block appear in the same segment. If that is the case, the assignment of occurrences of these tuples to parties will be independent. Let $t(i)$ be the number of segments in which i appears, and write $i \sim i'$ if tuples i and i' appear in the same segment at least once. Then the number of i' such that $i \sim i'$ is at most $t(i)kn/r$. For every $\ell = 1 \dots M$, we have

$$\Pr_{\mathcal{U}}[\exists i, i' \in U_\ell \text{ such that } i \sim i'] \leq \sum_i \sum_{i' \sim i} \left(\frac{1}{\frac{n}{4p}} \right)^2 \leq \sum_i \frac{t(i)kn}{r} \frac{16p^2}{n^2} = \frac{16p^2k^2n^2}{rn^2} = \frac{16p^2k^2}{r}.$$

Setting $r = 32k^2p^3$, the expected number of such blocks, conditioned on the event $\mathcal{E}_1$, is at most $\frac{N}{2p}$. Hence, by Markov's inequality,

$$\Pr_{\mathcal{U}}[\text{ the number of blocks with} \sim \text{tuples} \geq N/2 \mid \mathcal{E}_1] \leq \frac{1}{p}.$$

Let $\mathcal{E}_2$ be the event that the number of blocks **with no $\sim$ tuples** is at least $N' = \frac{N}{2} = \frac{n}{32p2^p}$. We have

$$\Pr_{\mathcal{U}}[\overline{\mathcal{E}_2} \mid \mathcal{E}_1] \leq \frac{1}{p}.$$

Now, we calculate the probability that the number of blocks with tuples that can be placed on the forehead of the p different parties is relatively large, conditioned on the events $\mathcal{E}_1$ and $\mathcal{E}_2$.

Given that a block has tuples occurring at most $2k$ times in s and no two tuples appear in the same segment, we calculate the probability that the assignment of segments to parties ensures that each of the tuples in the block can be placed on the forehead a different party. For a block U_ℓ, we construct a bipartite graph where the left vertices represent the p tuples in the block and the right vertices represent the p parties. We add an edge (i, j) if tuple i **cannot** be assigned to party j, because it is read in a segment in A_j. Observe that each tuple in block U_ℓ can be placed on the forehead of a different party if and only if this graph contains a perfect matching. Since the segments are assigned to the various A_j independently each with probability $1/p$, the resulting distribution on the graph is equivalent to that of the graph H in Lemma 3.3 with $d = 2k \leq \frac{1}{8} p \log p$.

For $\ell = 1, \dots, M$, let $X_\ell = 1$ iff the bipartite graph associated with block U_ℓ contains a perfect matching (i.e., each tuple in block U_ℓ can be placed on the forehead of a different party). Let $X = \sum_{\ell=1}^{N'} X_\ell$ be the number of such blocks from the N' blocks we obtain from conditioning on $\mathcal{E}_1$ and $\mathcal{E}_2$.
Since $p \geq 69$ we can apply Lemma 3.3 to obtain $\Pr_{\mathcal{U},\mathcal{A}}[X_\ell = 0] \leq \frac{1}{16}$. Using Markov's inequality, we calculate the probability that X is at least $N'/2$.

$$\Pr[X \leq N'/2 \mid \mathcal{E}_1, \mathcal{E}_2] \leq \frac{N'/16}{N'/2} = \frac{1}{8}.$$

Let $\mathcal{E}_3 \subseteq supp(\mathcal{U}) \times supp(\mathcal{A})$ be the event that there are at least $m = \frac{n}{64p2^p}$ blocks whose tuples can be placed on the foreheads of different parties. Combining all the above, $\mathcal{E}_3$ occurs except with probability at most $\epsilon_1 = e^{-\frac{n}{32p^2 4^p}} + \frac{1}{p} + \frac{1}{8}$ over the distributions $\mathcal{U}$ and $\mathcal{A}$. There must be some choice of $A = (A_1, \dots, A_p)$ for which the probability, over the distribution $\mathcal{U}$ on the grouping of blocks, that $\mathcal{E}_3$ does not occur is at most the average ϵ_1. We fix such an A.

Since the branching program B correctly computes $1GAP_n$ under distribution μ with probability at least $1 - \epsilon$, there must some choice U of grouping of tuples into blocks with $(U, A) \in \mathcal{E}_3$ such that B correctly computes $1GAP_n$ with probability at least $1-\epsilon-\epsilon_1$ under the distribution μ conditioned on the choice of U. (This conditional distribution is now determined entirely by the distribution $\mathcal{Z}$.) Let I, with $|I| = m$ be the set of blocks witnessing event $\mathcal{E}_3$. By simple averaging there must be some assignment ζ to the blocks not in I so that B correctly computes $1GAP_n$ with probability at least $1 - \epsilon - \epsilon_1$ under distribution μ conditioned on the choice of grouping U and assignment ζ. Let μ' be this conditional distribution.

By construction, we can create a p-partition $\mathcal{P}'$ of the set of $4pm$ nodes in the blocks in I so that each class contains all the nodes of precisely one tuple from each block. We extend $\mathcal{P}'$ to a partition $\mathcal{P}$ of all of $[n]$ by arbitrarily assigning to each class all the nodes from one tuple of each block not in I. Applying Proposition 3.2 with $r = 32k^2p^3$, we obtain the conclusion with distribution μ' and $\epsilon' = \epsilon + \epsilon_1$. □

3.3 Reducing GIP to $1GAP$

In [6], the authors give the following lower bound on the ϵ-error communication complexity of $GIP_{p,m}$ under the uniform distribution.

Proposition 3.6. *[6] $D^{uniform}_{\epsilon,p}(GIP_{p,m})$ is $\Omega(\frac{m}{4^p} + \log(1-2\epsilon))$.*

To obtain lower bounds on the distributional communication complexity of $1GAP$, we use a reduction from the distributional communication complexity of the generalized inner product problem $GIP_{p,m}$ where m is the value in the statement of Lemma 3.5.

Theorem 3.7. *Let p, μ', $\mathcal{P}$, and ϵ' be as in Lemma 3.5. Then $D^{\mu'}_{\epsilon',\mathcal{P}}(1GAP_n)$ is $\Omega(\frac{m}{4^p} + \log(1-2\epsilon'))$, where $m = \frac{n}{64p4^p}$.*

Proof. By construction, the distribution μ embeds the computation of an oblivious branching program $B_{GIPp,n}$ of width 4 and size $4pn + 2$ computing $GIP_{p,n}$ on a uniformly random input into the set of $1GAP_n$ instances. The branching program $B_{GIP_{p,n}}$ consists of n blocks of p layers each, plus a 0-node and a 1-node for the output.

We reduce the communication problem for $GIP_{p,m}$ under the uniform distribution to $1GAP_n$ under μ' by observing that a branching program $B_{GIP_{p,m}}$ for the $GIP_{p,m}$ problem on a uniformly random input can be embedded into the unfixed blocks of the $1GAP_n$ instances given by the distribution μ'. Let $z'_1, \ldots, z'_p$ be the remaining $4pm$ random choices given by $\mathcal{Z}'$, the unfixed portion $\mathcal{Z}'$ of $\mathcal{Z}$ under distribution μ'.

Let $n' = \frac{n}{4p} - m$. Note that the fixed portion of the assignment ζ can be viewed as the input to a $GIP_{p,n'}$ problem in a natural way. Observe that, by construction and the symmetry of $\oplus$, there is a path from nodes 1 to $n+1$ in the $1GAP_n$ instance given by μ' if and only if $GIP_{p,m}(z'_1, \ldots, z'_p) \oplus GIP_{p,n'}(\zeta) = 1$. Since ζ is fixed, $GIP_{p,n'}(\zeta)$ is known to all players and they can determine the value of $GIP_{p,m}(z'_1, \ldots, z'_p)$ by computing the $1GAP_n$ answer on the sample from μ' and XORing it with $GIP_{p,n'}(\zeta)$.

More precisely, the probability distribution μ' fixes the partition into blocks and the values of m' blocks of nodes. Let $U_{\ell_1}, \ldots, U_{\ell_m}$ be the remaining m blocks, ordered according to the fixed ordering in μ. The nodes of the i-th block of $B_{GIP_{p,m}}$ will correspond to the nodes of U_{ℓ_i} in the $1GAP_n$ problem in the obvious way. In order to solve $GIP_{p,m}$, the players apply the protocol for $1GAP_n$. Whenever they need to query the pointer for a node belonging to player i in a U_{ℓ_j} block they use the bit z'_{ij}. Whenever they need the pointer for any other node, the value is fixed by ζ so all players know it. Clearly μ' is induced by the uniform distribution over $\mathcal{Z}'$ and the partition induced on the inputs in $\mathcal{P}$ is precisely the standard partition on the inputs of $GIP_{p,m}$.

Therefore, we have a p-party protocol that computes $GIP_{p,m}$ for a $(1-\epsilon')$-fraction of the inputs under the uniform distribution on $(\{0,1\}^m)^p$. Hence, $D^{\mu'}_{\epsilon',\mathcal{P}}(1GAP_n) \geq D^{uniform}_{\epsilon',p}(GIP_{p,n})$ which is $\Omega(\frac{m}{4^p} + \log(1-2\epsilon'))$. $\square$

Finally, we obtain the desired time-space tradeoff.

Theorem 3.8. *Let $\epsilon < 1/2$. If a randomized oblivious branching program computes $1GAP_n$ with time T, space S and error at most ϵ, then $T = \Omega\left(n \log(\frac{n}{S}) \log\log(\frac{n}{S})\right)$.*

Proof. Let $\tilde{B}$ be such a randomized branching program computing $1GAP_n$. 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 ϵ of $\tilde{B}$ is $< 1/5$. Apply Proposition 3.1 to $\tilde{B}$ using distribution μ to obtain a deterministic oblivious branching program B with the same time and space bounds that computes $1GAP_n$ with error at most ϵ under μ.

Let $T = kn$ and let p be the smallest integer ≥ 69 such that $k \leq \frac{1}{16} p \log p$. 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 $\epsilon_1 = e^{-\frac{n}{32p^2 4^p}} + \frac{1}{p} + \frac{1}{8}$ which is $< 1/5$ for these values of p.

Since $69 \leq p \leq \sqrt{n}/2$ we can combine Lemma 3.5 and Theorem 3.7 to say that there is a constant C independent of n and p such that

$$32k^2p^3 \log W \geq C(\frac{m}{4^p} + \log(1-2\epsilon')),$$

where $m = \frac{n}{64p2^p}$ 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 $p^7 \log^2 pS \geq C_1 n 4^{-2p}$, for some constant C_1. Simplifying and taking logarithms, we have $p \geq C_2 \log(\frac{n}{S})$, for some constant C_2. Since p is the smallest integer ≥ 69 such that $k \leq \frac{1}{16} p \log p$, we have $k \geq C_3 \log(\frac{n}{S}) \log\log(\frac{n}{S})$ for some constant C_3 and the theorem follows. $\square$

4 Conclusion

We gave a $T = \Omega\left(n \log(\frac{n}{S}) \log\log(\frac{n}{S})\right)$ time-space tradeoff for the $1GAP$ problem on randomized oblivious branching programs, thus providing an $\Omega(\log(n/S) \log\log(n/S))$ factor separation from the power of general RAM algorithms. By the results of [9], we know an $\Omega(\log^2(n/S))$ gap deterministically and it would be interesting to match this gap in the case of randomized oblivious algorithms.

More broadly, it would be interesting to exploit the simulations in [3, 4, 11] of general RAMs by randomized oblivious ones to obtain larger time-space tradeoff lower bounds for other problems in NP. The known lower bounds for general branching programs in [8, 1, 2, 7] all require considerably more complicated arguments than used in our lower bound here. Those arguments are based on ideas, such as covering by embedded rectangles, that are inspired by communication complexity but are substantially more awkward to analyze. The simulations in [4, 11] show that to obtain a non-trivial time-space tradeoff of the form $TS = n^{1+\Omega(1)}$, it suffices to consider randomized oblivious algorithms. Though consideration of oblivious algorithms had already been a natural first step to try to obtain such results, these new simulations show that strong lower bounds for such algorithms would have implications for the general case.

References

[1] M. Ajtai. Determinism versus non-determinism for linear time RAMs with memory restrictions. *Journal of Computer and System Sciences*, 65(1):2–37, August 2002.

[2] M. Ajtai. A non-linear time lower bound for Boolean branching programs. *Theory of Computing*, 1(1):149–176, 2005.

[3] M. Ajtai. Oblivious RAMs without cryptographic assumptions. In *Proceedings of the Forty-Second Annual ACM Symposium on Theory of Computing*, pages 181–190, Cambridge, Ma, June 2010.

[4] M. Ajtai. Oblivious RAMs without cryptographic assumptions. Technical Report TR10-028, Electronic Colloquium in Computation Complexity, `http://www.eccc.uni-trier.de/eccc/`, 2010.

[5] Noga Alon and Wolfgang Maass. Meanders and their applications in lower bounds arguments. *Journal of Computer and System Sciences*, 37:118–129, 1988.

[6] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45(2):204–232, October 1992.

[7] P. Beame, M. Saks, X. Sun, and E. Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *Journal of the ACM*, 50(2):154–195, 2003.

[8] Paul Beame, T. S. Jayram, and Michael Saks. Time-space tradeoffs for branching programs. *Journal of Computer and System Sciences*, 63(4):542–572, December 2001.

[9] Paul Beame and Erik Vee. Time-space tradeoffs, multiparty communication complexity, and nearest-neighbor problems. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 688–697, Montreal, Quebec, Canada, May 2002.

[10] R. E. Bryant. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):283–316, 1992.

[11] I. Damgård, S. Meldgaard, and J. B. Nielsen. Perfectly secure oblivious RAM without random oracles. Technical Report 2010/108, Cryptology ePrint Archive, 2010.

[12] O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 182–194, New York, NY, May 1987.

[13] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[14] R. Ostrovsky. Efficient computation on oblivious RAMs. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 514–523, Baltimore, MD, May 1990.

[15] M. Sauerhoff. A lower bound for randomized read-k-times branching programs. In *(STACS) 98: 15th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*, pages 105–115, Paris, France, February 1998. Springer-Verlag.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de