The Random-Query Model and the Memory-Bounded Coupon Collector

Ran Raz
Department of Computer Science, Princeton University, United States
ranr@cs.princeton.edu

Wei Zhan
Department of Computer Science, Princeton University, United States
weizhan@cs.princeton.edu

Abstract

We study a new model of space-bounded computation, the *random-query* model. The model is based on a branching-program over input variables $x_1, \ldots, x_n$. In each time step, the branching program gets as an input a random index $i \in \{1, \ldots, n\}$, together with the input variable x_i (rather than querying an input variable of its choice, as in the case of a standard (oblivious) branching program). We motivate the new model in various ways and study time-space tradeoff lower bounds in this model.

Our main technical result is a quadratic time-space lower bound for zero-error computations in the random-query model, for XOR, Majority and many other functions. More precisely, a zero-error computation is a computation that stops with high probability and such that conditioning on the event that the computation stopped, the output is correct with probability 1. We prove that for any Boolean function $f : \{0,1\}^n \to \{0,1\}$, with sensitivity k, any zero-error computation with time T and space S, satisfies $T \cdot (S + \log n) \geq \Omega(n \cdot k)$. We note that the best time-space lower bounds for standard oblivious branching programs are only slightly super linear and improving these bounds is an important long-standing open problem.

To prove our results, we study a memory-bounded variant of the coupon-collector problem that seems to us of independent interest and to the best of our knowledge has not been studied before. We consider a zero-error version of the coupon-collector problem. In this problem, the coupon-collector could explicitly choose to stop when he/she is sure with zero-error that all coupons have already been collected. We prove that any zero-error coupon-collector that stops with high probability in time T, and uses space S, satisfies $T \cdot (S + \log n) \geq \Omega(n^2)$, where n is the number of different coupons.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases random-query model, time-space trade-offs, branching programs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.20

Funding *Ran Raz*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grant No. CCF-1714779.
Wei Zhan: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grant No. CCF-1714779.

1 Introduction

In this paper, we introduce a new model for studying time-space tradeoff lower bounds for computation, the *random-query* model. The model is based on a *branching program*. Roughly speaking, a branching program of length T and width 2^S, over input variables $x_1, \ldots, x_n$, is a directed (multi) graph with vertices arranged in $T + 1$ layers containing at most 2^S vertices each. Intuitively, each layer represents a time step and each vertex represents a memory state of the program. In layer-0 of the program, there is only one vertex, called the start vertex. Each leaf of the program is labelled by an element from $\{0,1\}$ that we think of as the output of the program on that leaf.

In a standard branching program, every non-leaf vertex v in the program is labeled by an input variable x_v and has 2 outgoing edges, labeled by 0 and 1, going into vertices in the next layer. Intuitively, x_v is the input variable read by the vertex v. The program is called *oblivious* if all the vertices in the same layer read the same input variable. Given a branching program, the input $x_1, \dots, x_n$ defines a computation-path, by starting from the start vertex and following in each step the edge labeled by the value of the corresponding input variable. The program outputs the label of the leaf reached by the computation path.

In the random-query model, every non-leaf vertex v in the program has $2n$ outgoing edges, labeled by each element of $\{1, \dots, n\} \times \{0, 1\}$ exactly once. Given such a program and input $x_1, \dots, x_n$, the computation-path starts from the start vertex and follows in each step the edge labeled by (i, x_i), where $i \in \{1, \dots, n\}$ is random. (Intuitively, the program reads a random index $i \in \{1, \dots, n\}$, together with the input variable x_i). As before, the program outputs the label of the leaf reached by the computation path.

1.1 Motivation

We have various motivations to study the new model. First, it seems to us an interesting model in its own right. The standard model of space-bounded computation is not always fully convincing in all settings, as it is not clear why would a machine be able to store for free the n input variables, while at the same time have a very restricted (typically, of size much smaller than n) additional memory. Moreover, in various situations the random-query model seems to be the natural one to use. Consider for example the following situation: you are at a party and you want to know if the majority of the participants prefer coffee or tea. (Assume that you know the number of participants in the party, that that number is odd and that you know all participants (or they are labeled $1, \dots, n$)). Assume that at each time step you meet a random participant and she/he tells you their preference. How long would it take to figure out if the majority prefers coffee or tea if your memory is bounded? Another example may be a distributed setting where n players have one input variable each and they keep sending these input variables to a central player who needs to compute a Boolean function of all of them. However, the input variables arrive to the central player in an arbitrary order.

Second, we study time-space lower bounds for the random-query model in order to make progress in proving time-space lower bounds for standard (oblivious) branching programs. Time-space lower bounds for branching programs have been studied in numerous works (see for example [3, 1, 2, 5, 6]). Currently, the best time-space lower bounds for any explicit function are only slightly super linear and improving these lower bounds has been a very important and long standing open problem in computational complexity. In section 5, we show that various extensions of our results would imply such strong time-space lower bounds. Roughly speaking, our time-space lower bounds for the random-query model are proved for the case where the indices $i_1, i_2, i_3, \dots$ of the input variables read by the program at time steps $1, 2, 3, \dots$ are mutually independent random variables, while in order to extend these lower bounds to standard branching programs one needs to generalize the proofs to the case where some of these indices are known to be the same. Interestingly, the key component of our proof, Theorem 2, does apply to the more general case where some of the indices are known to be the same. However, the main results do not.

Third, the new model is related to several other problems that have been studied recently. First, it is related to the recent line of works on proving time-space lower bounds for learning (see for example [16, 18, 14, 11, 12, 15, 13, 4, 8, 7, 17, 9]). Indeed, computing a function $f : \{0, 1\}^n \to \{0, 1\}$ in the random-query model is equivalent to the task of distinguishing between the following two families of distributions (which is a learning task[1]): For $x \in \{0,1\}^n$, let $\mathcal{D}_x$ be the distribution of the random variable (i, x_i), where $i \in \{1,\dots,n\}$ is uniformly distributed. The task is to distinguish between a distribution taken from $\{\mathcal{D}_x\}_{x:f(x)=0}$ and a distribution taken from $\{\mathcal{D}_x\}_{x:f(x)=1}$, from a stream of independent samples. Second, the random-query model is similar to a recently studied model of streaming complexity, where a source of i.i.d samples of edges of a graph is considered [10]. In particular, [10] studied approximation algorithms for the maximum matching problem in that model. The main difference from our model is that they studied the space needed for approximate computation in the case where the number of samples is smaller than the number of input variables, while we study time-space tradeoffs for exact computation in the case where the number of samples may be much larger than the number of input variables.

Finally, it turns out that in the zero-error case, the random-query model is closely related to a memory-bounded variant of the coupon-collector problem, a problem that seems to be of independent interest and to the best of our knowledge has not been studied before. In our variant of the problem, the coupon collector gets a stream of random elements from the set $\{1,\dots,n\}$ and needs to stop when she is sure with zero-error that all elements of $\{1,\dots,n\}$ have already passed. The question is what is the time T needed when the memory size of the coupon collector is bounded by S.

1.2 Our Results

In Theorem 5, we prove that any algorithm for the zero-error coupon-collector problem that runs in time T and space S satisfies $T \cdot (S + \log n) \geq \Omega(n^2)$. This result is essentially tight. In Theorem 6, we prove that in the random-query model, any zero-error computation of XOR or Majority (or any other function with sensitivity $\Omega(n)$) that runs in time T and space S satisfies $T \cdot (S + \log n) \geq \Omega(n^2)$. The results for XOR and Majority are essentially tight (See Remarks 8 and 9 for the discussions on tightness). More generally, in the random-query model, any zero-error computation of a function with sensitivity k that runs in time T and space S satisfies $T \cdot (S + \log n) \geq \Omega(n \cdot k)$.

A very interesting open problem is to prove similar time-space lower bounds for the random-query model in the bounded-error case, rather than the zero-error case (Conjecture 1).

In Theorem 2, we prove time-space lower bounds for a special type of branching programs called set-labeled branching programs, in the random-query model. Intuitively, a set-labeled branching program is a branching-program for the coupon-collector problem, such that each vertex in the program "remembers" a set of coupons that must have been collected if that vertex was reached.

1.3 Paper Organization

The paper is organized as follows. In section 3, we prove the tight time-space lower bound for set-labeled branching programs, in the random-query model. In section 4, we reduce zero-error computation tasks in the random-query model, including the coupon-collector problem and function evaluation, to set-labeled branching programs, and hence prove tight time-space lower bounds for both problems. In section 5, we illustrate how lower bounds in the random-query model with special input distribution imply lower bounds for oblivious branching programs.

[1] Technically it is a testing task, which is easier than learning.

2 Preliminaries

For an integer n, we use $[n]$ to denote $\{1, 2, \ldots, n\}$. For any set A and an n-tuple $x \in A^n$, we use x_i to denote the i-th element of x. For any $x \in \{0,1\}^n$, let $x^{(i)}$ be the vector that is the same as x but with the i-th coordinate flipped. Given a boolean function $f : \{0,1\}^n \to \{0,1\}$, let $s(f,x)$ be the sensitivity of f at x, that is the number of coordinates $i \in [n]$ such that $f(x^{(i)}) \neq f(x)$, and let $s(f) = \max_x s(f,x)$ be the sensitivity of f.

2.1 Coupon-Collector Problem

The classical coupon-collector problem asks how large T should be, so that a uniformly random T-tuple in $[n]^T$ contains every element of $[n]$ with high probability. Generalizing the goal to a subset $A \subseteq [n]$, we have the following answer:

▶ **Proposition 1.** *Given any subset $A \subseteq [n]$, for a uniformly random $i \in [n]^T$, the probability that $A \nsubseteq \{i_1, \ldots, i_T\}$ is at most $n(1 + \log|A|)T^{-1}$.*

The proof follows directly from the fact that the expected waiting time for every element in A to appear is $n \sum_{j=1}^{|A|} j^{-1} \geq n(1 + \log|A|)$, and Markov's inequality.

In this paper, we consider a zero-error version of the coupon-collector problem. In this problem, the coupon collector could explicitly choose to stop when she is sure with zero-error that every element in A has already been collected. The results in this paper show that with bounded memory, the zero-error coupon-collector cannot stop within few (say, $O(n \log|A|)$) turns with high probability, in contrast to the proposition above.

2.2 Random-Query Model

In the random-query model, at each step $t \in \mathbb{N}_+$ a uniformly random index $i_t \in [n]$ is provided. When the problem specifies an input $x \in \{0,1\}^n$, at each step t the value of the bit $x_{i_t} \in \{0,1\}$ is also given along with the random index i_t. In this paper, we consider two cases for the joint distribution of the indices:

Independent The indices $i_1, i_2, \ldots$ are mutually independent.

Recurring The only dependencies allowed among $i_1, i_2, \ldots$ are equalities. More formally, there is a partition $p : \mathbb{Z}_+ \to \mathbb{Z}_+$, such that $i_t = i'_{p(t)}$ for every $t \in \mathbb{Z}_+$, where $i'_1, i'_2, \ldots$ are mutually independent and uniformly random over $[n]$.

For the rest of the paper, we refer to the two cases as *independent distribution* and *recurring distributions*. Notice that the independent distribution is a special case of the recurring ones. The recurring distributions are closely related to oblivious branching programs; see Section 5 for a detailed discussion.

2.3 Computational Models

The computational models we consider are based on branching programs. A *branching program* of length T and width 2^S is a directed (multi) graph with vertices arranged in $T+1$ layers containing at most 2^S vertices each. Denote the set of vertices in the i-th layer by $\mathcal{L}_i$, for $i = 0, 1, \ldots, T$. In $\mathcal{L}_0$ there is only one vertex, called the *start vertex*. Every vertex in $\mathcal{L}_T$ has out-degree 0, and is called a *leaf*. The outgoing edges from every non-leaf vertex in $\mathcal{L}_i$ only go to vertices in $\mathcal{L}_{i+1}$, for every $i < T$.

A simple branching program is one such that every non-leaf vertex has n outgoing edges, labeled with each element in $[n]$ exactly once. We consider two types of simple branching programs:

- A set-labeled branching program is a simple branching program, where every vertex v is labeled with a set $H(v) \subseteq [n]$, satisfying the following *soundness* condition: if an edge from vertex u to vertex v is labeled with $i \in [n]$, it must hold that $H(v) \subseteq H(u) \cup \{i\}$. The start vertex must be labeled with $\varnothing$.
- A branching program for the coupon-collector problem is a simple branching program such that every leaf is labeled with either 'accept' or 'reject'.

When the indices $i_1, \ldots, i_T \in [n]$ are given, the computation path in a simple branching program starts from the start vertex, and at step t follows the edge labeled with i_t until reaching a leaf v, and outputs the label of v.

Given a function $f : \{0,1\}^n \to \{0,1\}$, a branching program computing f is one such that every non-leaf vertex has $2n$ outgoing edges, labeled with each element in $[n] \times \{0,1\}$ exactly once. Every leaf v in the program is labeled with an output $\tilde{f}_v \in \{0, 1, \Box\}$. When an input $x \in \{0,1\}^n$ and the indices $i_1, \ldots, i_T \in [n]$ are given, the computation path in the branching program starts from the start vertex, and at step t follows the edge labeled with (i_t, x_{i_t}) until reaching a leaf v, and outputs $\tilde{f}_v$.

In the random-query model where the indices $i_1, \ldots, i_T$ are given according to a specified distribution, we define the success of every type of branching program as follows:

- We say that a set-labeled branching program *succeeds on* $A \subseteq [n]$, if the probability that the output of the branching program $H(v) \supseteq A$ is at least $1/2$.
- For the coupon-collector problem, we say the branching program *collects* $A \subseteq [n]$ *with zero-error*, if the probability that the branching program outputs 'accept' is at least $1/2$, and conditioned on outputting 'accept', the probability that $\{i_1, \ldots, i_T\} \supseteq A$ is 1.
- For computing a function f, we say that the branching program *computes* f *with error* ε, if for every $x \in \{0,1\}^n$, the probability that the output of the branching program $\tilde{f}_v = f(x)$ is at least $1 - \varepsilon$. We say that the branching program *computes* f *with zero-error*, if for every $x \in \{0,1\}^n$, the probability that the output of the branching program $\tilde{f}_v \in \{0,1\}$ is at least $1/2$, and the probability that $\tilde{f}_v = 1 - f(x)$ is zero.

3 Lower Bounds for Set-Labeled Branching Programs

In this section, we prove the following theorem:

▶ **Theorem 2.** *Under the random-query model with any recurring distribution, for any set $A \subseteq [n]$, any set-labeled branching program of width $2^S \geq |A|$ that succeeds on A must have length at least $\frac{n|A|}{8S}$ for sufficiently large n.*[2]

Fix such a set-labeled branching program. We first prove an upper bound on the probability of the computation path reaching two given vertices:

▶ **Lemma 3.** *For any two vertices u, v in a set-labeled branching program, where $u \in \mathcal{L}_i$, $v \in \mathcal{L}_j$ and $i < j$. Under the random-query model with any recurring distribution,*

$$\Pr[\textit{reaching } u \wedge \textit{reaching } v] \leq \left(\frac{j-i}{n}\right)^{|H(v) \setminus H(u)|}.$$

[2] Notice that by definition, a branching program of width $2^S < |A|$ is also a branching program of width $|A|$. Therefore for smaller widths, the theorem still holds, but with an additional $log|A|$ overhead on S. Theorems 5 and 6 work similarly.

Proof. Let $p : \mathbb{Z}_+ \to \mathbb{Z}_+$ be the partition for the recurring distribution. Let $\ell = |\{p(k) \mid i < k \le j\}|$. The indices received from the random queries between layer i and layer j are uniformly distributed over $[n]^\ell$. Let G be the random variable that represents the set of indices received between layer i and layer j. By the soundness requirement of set-labeled branching programs, if the computation path reaches u and then v, the set G corresponding to this path must satisfy $H(v) \subseteq H(u) \cup G$. Therefore,

$$\Pr[\text{reaching } u \wedge \text{reaching } v] \le \Pr[H(v) \subseteq H(u) \cup G] = \Pr[H(v) \setminus H(u) \subseteq G].$$

If $\ell < |H(v) \setminus H(u)|$ then the above probability is zero. Otherwise by (over)counting the positions where the elements of $|H(v) \setminus H(u)|$ appear and the union bound we have

$$\begin{aligned}\Pr[H(v) \setminus H(u) \subseteq G] &\le \frac{\ell!}{(\ell - |H(v) \setminus H(u)|)!} \cdot n^{-|H(v)\setminus H(u)|} \\ &\le \left(\frac{\ell}{n}\right)^{|H(v)\setminus H(u)|} \\ &\le \left(\frac{j-i}{n}\right)^{|H(v)\setminus H(u)|}.\end{aligned}$$

◀

▶ Remark 4. For the independent distribution, the above argument yield:

$$\Pr[\text{reaching } v \mid \text{reaching } u] \le \left(\frac{j-i}{n}\right)^{|H(v)\setminus H(u)|}.$$

The weaker result in Lemma 3, however, holds more generally for any recurring distribution. It is also strong enough for proving Theorem 2.

Proof for Theorem 2. Suppose the length of the set-labeled branching program is T. Define the weight of a vertex v as $W(v) = \Pr[\text{reaching } v]$. For a set of vertices $\mathcal{A}$, let $W(\mathcal{A}) = \sum_{v\in\mathcal{A}} W(v)$. Since the leaves are all in $\mathcal{L}_T$, for every $0 \le i \le T$ we have $W(\mathcal{L}_i) = 1$. The fact that the branching program succeeds on $A \subseteq [n]$ translates to:

$$\sum_{\substack{v\in\mathcal{L}_T \\ A\subseteq H(v)}} W(v) \ge 1/2. \tag{1}$$

We divide the branching program into $\frac{|A|}{2S}$ stages, each consists of a consecutive part of the layers. For every $0 \le k \le \frac{|A|}{2S}$, let i_k be the smallest index of a layer $\mathcal{L}_i$ such that

$$\sum_{\substack{v\in\mathcal{L}_i \\ |H(v)|\ge 2kS}} W(v) \ge \frac{kS}{|A|}.$$

By (1) we know such a layer must exist. Now the k-th stage consists of the layers from $\mathcal{L}_{i_k}$ to $\mathcal{L}_{i_{k+1}-1}$. Let

$$\mathcal{A}_k = \{u \in \mathcal{L}_{i_k} \mid |H(u)| \ge 2kS\}, \qquad \mathcal{B}_k = \{u \in \mathcal{L}_{i_k-1} \mid |H(u)| < 2kS\}.$$

By the definitions of i_k, we know that $W(\mathcal{A}_k) \ge kS/|A|$, $W(\mathcal{B}_k) > 1 - kS/|A|$.

Now we show that every stage contains at least $(n/3-1)$ layers. Suppose for contradiction that for some k, it holds that $i_{k+1} - i_k < n/3 - 1$. For any two vertices $u \in \mathcal{B}_k$ and $v \in \mathcal{A}_{k+1}$, by Lemma 3 we have

$$\Pr[\text{reaching } u \wedge \text{reaching } v] \leq \left(\frac{i_{k+1} - i_k + 1}{n}\right)^{|H(v) \setminus H(u)|} < 3^{-2S}.$$

Therefore, applying the union bound gives:

$$\begin{aligned}
&\Pr[\text{reaching } \mathcal{L}_{i_k - 1} \wedge \text{reaching } \mathcal{L}_{i_{k+1}}] \\
\leq &\Pr[\text{reaching } \mathcal{L}_{i_k - 1} \setminus \mathcal{B}_k] + \Pr[\text{reaching } \mathcal{L}_{i_{k+1}} \setminus \mathcal{A}_{k+1}] + \Pr[\text{reaching } \mathcal{B}_k \wedge \text{reaching } \mathcal{A}_{k+1}] \\
\leq &1 - W(\mathcal{B}_k) + 1 - W(\mathcal{A}_{k+1}) + \sum_{\substack{u \in \mathcal{B}_k \\ v \in \mathcal{A}_{k+1}}} \Pr[\text{reaching } u \wedge \text{reaching } v] \\
< &\frac{kS}{|A|} + 1 - \frac{(k+1)S}{|A|} + 2^S \cdot 2^S \cdot 3^{-2S} < 1.
\end{aligned}$$

The second last step is because there are at most 2^S vertices in each layer, and the last step is because $2^S \geq |A|$. However, since the computation path must pass through both $\mathcal{L}_{i_k - 1}$ and $\mathcal{L}_{i_{k+1}}$, the probability above must be 1, which is a contradiction.

Thus we conclude that, for n large enough, $i_{k+1} - i_k \geq n/3 - 1 \geq n/4$. Therefore,

$$T \geq \sum_{0 \leq k < |A|/2S} (i_{k+1} - i_k) \geq \frac{n|A|}{8S}.$$

◀

4 Lower Bounds for Zero-error Computations under Independent Distribution

▶ **Theorem 5.** *Under the random-query model with the independent distribution, for any set $A \subseteq [n]$, any branching program for the coupon-collector problem of width $2^S \geq |A|$ which collects A with zero-error must have length at least $\frac{n|A|}{8S}$ for sufficiently large n.*

Proof. We show that for such a branching program, we can assign each vertex v with a label $H(v) \subseteq [n]$ so that the branching program is set-labeled. Let $P(v)$ be the collection of directed paths from the starting vertex to v. For every directed path p let $h(p)$ be the collection of indices labeled on the edges of p. Then we define $H(v) = \cap_{p \in P(v)} h(p)$.

The starting vertex is clearly labeled with the empty set. To check the soundness, consider an edge e from vertex u to vertex v labeled with i. For every path $p \in P(u)$, the concatenation pe is a path in $P(v)$, and $h(pe) = h(p) \cup \{i\}$. Therefore $H(v) \subseteq \cap_{p \in P(u)} h(pe) = H(u) \cup \{i\}$.

Notice that every path from the starting vertex to a leaf corresponds to a collection of indices $i_1, \ldots, i_T$, that are given with probability $n^{-T} > 0$ under the independent distribution. Since the branching program collects A with zero-error, for every path to an 'accept' leaf it must holds $A \subseteq \{i_1, \ldots, i_T\}$, so every 'accept' leaf v is now labeled with $H(v) \supseteq A$. Therefore, as a set-labeled branching program it succeeds on A. By Theorem 2 we know the length of the branching program is at least $\frac{n|A|}{8S}$ for sufficiently large n. ◀

▶ **Theorem 6.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function with sensitivity $s(f)$. Under the random-query model with the independent distribution, any branching program of width $2^S \geq n$ which computes f with zero-error must have length at least $\frac{n \cdot s(f)}{8S}$ for sufficiently large n.*

Proof. Suppose there is a branching program $\mathcal{P}$ of width 2^S and length T that computes f with zero-error. Let $x \in \{0,1\}^n$ be an input such that $s(f) = s(f,x)$, and let $A = \{i \in [n] \mid f(x) \neq f(x^{(i)})\}$. We show below that from $\mathcal{P}$, one can extract a simple branching program $\mathcal{P}'$ for the coupon-collector problem of width at most 2^S and length T, which collects A with zero-error. Since $|A| = s(f)$, by Theorem 5 we know $T \geq \frac{n \cdot s(f)}{8S}$ for sufficiently large n.

We construct $\mathcal{P}'$ inductively to simulate $\mathcal{P}$ on input x. For vertex v in $\mathcal{P}$ we use v' to denote its corresponding vertex in $\mathcal{P}'$. The start vertex v_0' in $\mathcal{P}'$ corresponds to the start vertex v_0 in $\mathcal{P}$. If in $\mathcal{P}$ there exists an edge from u to v labeled with (i, x_i), and u' is in $\mathcal{P}'$, then add v' to $\mathcal{P}'$ (if v' is not already there), and add an edge from u' to v' labeled with i. Finally, for every leaf v' in $\mathcal{P}'$, label v' with 'accept' if $\tilde{f}_v = f(x)$, otherwise label v' with 'reject'.

First notice that under the independent distribution, the probability of reaching a vertex v' in $\mathcal{P}'$ is exactly the same as the probability of reaching v in $\mathcal{P}$ with the input x. Since the probability that $\mathcal{P}$ outputs $f(x)$ on input x is at least $1/2$, the probability that $\mathcal{P}'$ outputs 'accept' is also at least $1/2$.

We now show that conditioned on reaching a leaf v' in $\mathcal{P}'$ labeled with 'accept', it must hold that $A \subseteq \{i_1, \ldots, i_T\}$. Suppose not, then for some index $i \in A$ there is a path p' from the start vertex to v' where no edge is labeled with i. Consider the corresponding path p in $\mathcal{P}$. On input $x^{(i)}$, the computation follows the path p with non-zero probability and outputs $\tilde{f}_v = f(x) \neq f(x^{(i)})$, which contradicts the zero-error property of $\mathcal{P}$. That concludes the proof that $\mathcal{P}'$ collects A with zero-error. ◀

For the large class of functions with sensitivity $\Omega(n)$, Theorem 6 provides the quadratic time-space lower bound:

▶ **Corollary 7.** *Let f be a boolean function on n-bits with sensitivity $\Omega(n)$ (For instance, AND, XOR, Majority, s-t connectivity, etc.). Under the random-query model with the independent distribution, any branching program of width $2^S \geq n$ which computes f with zero-error must have length $\Omega(n^2/S)$.*

▶ Remark 8. Theorem 6 is tight up to logarithmic factors, in the sense that for every $m \leq n$, the function $x_1 \oplus \cdots \oplus x_m$ can be computed with zero-error within S space and $O(nmS^{-1}\log n)$ steps. We briefly sketch the algorithm here: Equally partition $[m]$ into $O(mS^{-1})$ parts, each of size $O(S)$. For each part P, use $O(n \log n)$ steps to record the values x_i for all indices $i \in P$. If any $i \in P$ does not appear within these $O(n \log n)$ steps, output $\square$. Otherwise compute the partial parity $\bigoplus_{i \in P} x_i$, and accumulate the partial parities.

As the lower bound in Theorem 6 is derived directly from Theorem 5 and further from Theorem 2, variants of the above algorithm also imply that Theorems 2 and 5 are tight up to logarithmic factors.

Similar to the case in the coupon-collector problem, the zero-error guarantee is crucial to Theorem 6, since for instance, the n-bit AND function can be computed with constant error by a branching program of length $O(n)$ and width $O(1)$. However, when specified to the parity function, the best trade-off seems to be still quadratic even in the bounded-error setting. We propose the following conjecture:

▷ Conjecture 1. Under the random-query model with the independent distribution, any branching program of length T and width 2^S which computes $x_1 \oplus \cdots \oplus x_n$ with error $1/3$ must satisfy $TS = \widetilde{\Omega}(n^2)$.

▶ Remark 9. Besides the algorithm mentioned above, there is another essentially different algorithm for computing parity (which actually computes the Hamming weight) with bounded error: Equally partition $[n]$ into $O(S/\log n)$ parts. For each part P, record the number of steps t when a pair (i, x_i) such that $i \in P$ and $x_i = 1$ is received, and finally approximate the partial sum $\sum_{i\in P} x_i$ with the integer closest to tn/T. By Chernoff bound, $T = O(n^2 S^{-1} \log^2 n)$ is enough so that the approximation of each part is wrong with probability $O(n^{-1})$.

Notice that this algorithm does not work in the zero-error setting. While the previous algorithm corresponds directly to a set-labeled branching program, it is not clear whether this approximation algorithm is related to set-labeled branching programs or not.

5 Oblivious Branching Programs and Random-Query Model

The random input model with recurring distributions is closely related to oblivious branching programs. In this section, we present two potential directions to prove strong lower bounds for oblivious branching programs, both via proving lower bounds in the random-query model. Let $\mathrm{SURJ}_{n,m} : [n]^m \to \{0, 1\}$ be the surjectivity function: $\mathrm{SURJ}_{n,m}(i) = 1$ if and only $\{i_1, \dots, i_m\} = [n]$.

▶ **Theorem 10.** *For any $m \geq 2n(\log n + 1)$, any deterministic oblivious branching program computing $\mathrm{SURJ}_{n,m}$ is also a branching program for the coupon-collector problem that collects $[n]$ with zero-error under some recurring distribution.*

Proof. Suppose at level $t - 1$ the oblivious branching program reads $i_{p(t)}$, for some function $p : \mathbb{Z}_+ \to [m]$. Use p as the partition in the recurring distribution in the random-query model, then the computation of the branching program for the coupon-collector problem is exactly the same as in the oblivious branching program with a uniformly random input $i \in [n]^m$. Proposition 1 shows that the probability of $\mathrm{SURJ}_{n,m}(i) = 1$ is at least $1/2$. As the deterministic oblivious branching program always outputs correctly, as a branching program for the coupon-collector problem it succeeds with zero-error. ◀

For any function $f : \{0, 1\}^n \to \{0, 1\}$ and $m \geq n$, let $f^* : [n]^m \times \{0, 1\}^m \to \{0, 1\}$ be a partial function defined as follows: $f^*(i, y)$ is well-defined for $i \in [n]^m$ and $y \in \{0, 1\}^m$, if and only if $\mathrm{SURJ}_{n,m}(i) = 1$, and whenever $i_j = i_k$ it must hold $y_j = y_k$. When $f^*(i, y)$ is well-defined, the value of $f^*(i, y)$ is $f(y_{j_1}, \dots, y_{j_n})$, where for every $\iota \in [n]$, j_ι is some $j \in [m]$ such that $i_j = \iota$.

▶ **Theorem 11.** *Given any function $f : \{0, 1\}^n \to \{0, 1\}$. For any $m \geq 3n(\log n + 1)$, if there is a deterministic oblivious branching program computing f^* of length T and width 2^S (on the inputs where f^* is well-defined), then there is a branching program of the same length and width, that computes f with error $1/3$ in the random-query model under some recurring distribution.*

Proof. Add dummy levels to the oblivious branching program to double the length, such that if originally at level t the branching program reads either i_j or y_j, now it reads i_j at level $2t$ and y_j at level $2t + 1$. The oblivious branching program now can be regarded as the one of length T and width 2^S that at each level t reads a pair $(i_{p(t)}, y_{p(t)})$, for some function $p : \mathbb{Z}_+ \to [m]$. Use p as the partition in the recurring distribution. For any fixed $x \in \{0, 1\}^n$, the computation in the random-query model on input x is exactly the same as in the oblivious branching program with a uniformly random $i \in [n]^m$, and input $y \in \{0, 1\}^m$ defined as $y_j = x_{i_j}$. For such i and y, $f^*(i, y)$ is well-defined if and only if $\mathrm{SURJ}_{n,m}(i) = 1$, and Proposition 1 indicates the probability that $f^*(i, y)$ is well-defined is at least $2/3$. Since whenever $f^*(i, y)$ is well-defined, the deterministic oblivious branching program correctly outputs $f(y_{j_1}, \dots, y_{j_n}) = f(x)$, as a branching program under the random-query model it computes f with error $1/3$. ◀

As a corollary, if in the random-query model we were able to prove a time-space lower bound that holds under any recurring distribution, either for the zero-error coupon-collector problem, or for any bounded-error computation, we would immediately have the same lower bound (up to logarithmic factors) on deterministic oblivious branching programs.

References

1 Miklós Ajtai. Determinism versus non-determinism for linear time rams (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 632–641, 1999. doi:10.1145/301250.301424.

2 Miklós Ajtai. A non-linear time lower bound for boolean branching programs. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 60–70, 1999. doi:10.1109/SFFCS.1999.814578.

3 László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992. doi:10.1016/0022-0000(92)90047-M.

4 Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 843–856, 2018.

5 Paul Beame, T. S. Jayram, and Michael E. Saks. Time-space tradeoffs for branching programs. *J. Comput. Syst. Sci.*, 63(4):542–572, 2001. doi:10.1006/jcss.2001.1778.

6 Paul Beame, Michael E. Saks, Xiaodong Sun, and Erik Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195, 2003. doi:10.1145/636865.636867.

7 Yuval Dagan and Ohad Shamir. Detecting correlations with little memory and communication. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 1145–1198, 2018.

8 Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 990–1002, 2018. doi:10.1145/3188745.3188962.

9 Sumegha Garg, Ran Raz, and Avishay Tal. Time-space lower bounds for two-pass learning. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, pages 22:1–22:39, 2019. doi:10.4230/LIPIcs.CCC.2019.22.

10 Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. *CoRR*, abs/1907.05725, 2019.

11 Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1067–1080, 2017. doi:10.1145/3055399.3055430.

12 Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1516–1566, 2017.

13 Dana Moshkovitz and Michal Moshkovitz. Entropy samplers and strong generic lower bounds for space bounded learning. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 28:1–28:20, 2018. doi:10.4230/LIPIcs.ITCS.2018.28.

14 Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 266–275, 2016. doi:10.1109/FOCS.2016.36.

15 Ran Raz. A time-space lower bound for a large class of learning problems. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 732–742, 2017. doi:10.1109/FOCS.2017.73.

16 Ohad Shamir. Fundamental limits of online and distributed algorithms for statistical learning and estimation. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 163–171, 2014.

17 Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 890–901, 2019. doi:10.1145/3313276.3316403.

18 Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 1490–1516, 2016.