Lower Bounds on Black-Box Reductions of Hitting to Density Estimation

Roei Tell
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel
roei.tell@weizmann.ac.il

Abstract

Consider a deterministic algorithm that tries to find a string in an unknown set $S \subseteq \{0,1\}^n$, under the promise that S has large density. The only information that the algorithm can obtain about S is *estimates of the density of* S in adaptively chosen subsets of $\{0,1\}^n$, up to an additive error of $\mu > 0$. This problem is appealing as a derandomization problem, when S is the set of satisfying inputs for a circuit $C : \{0,1\}^n \to \{0,1\}$ that accepts many inputs: In this context, an algorithm as above constitutes a deterministic black-box reduction of the problem of *hitting* C (i.e., finding a satisfying input for C) to the problem of *approximately counting* the number of satisfying inputs for C on subsets of $\{0,1\}^n$.

We prove tight lower bounds for this problem, demonstrating that naive approaches to solve the problem cannot be improved upon, in general. First, we show a tight trade-off between the estimation error μ and the required number of queries to solve the problem: When $\mu = O(\log(n)/n)$ a polynomial number of queries suffices, and when $\mu \geq 4 \cdot (\log(n)/n)$ the required number of queries is $2^{\Theta(\mu \cdot n)}$. Secondly, we show that the problem "resists" parallelization: Any algorithm that works in iterations, and can obtain $p = p(n)$ density estimates "in parallel" in each iteration, still requires $\Omega\left(\frac{n}{\log(p)+\log(1/\mu)}\right)$ iterations to solve the problem.

This work extends the well-known work of Karp, Upfal, and Wigderson (1988), who studied the setting in which S is only guaranteed to be non-empty (rather than dense), and the algorithm can only probe subsets for the *existence* of a solution in them. In addition, our lower bound on parallel algorithms affirms a weak version of a conjecture of Motwani, Naor, and Naor (1994); we also make progress on a stronger version of their conjecture.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness, Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Approximate Counting, Lower Bounds, Derandomization, Parallel Algorithms, Query Complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.58

Related Version A full version of this paper is available on ECCC [8].

Funding This research was partially supported by the Minerva Foundation with funds from the Federal German Ministry for Education and Research.

Acknowledgements The author thanks his advisor, Oded Goldreich, for clearly defining the problem studied in this paper, and for many useful discussions and ideas. The author also thanks Karthik C. S. for useful discussions and for many valuable comments that improved the presentation of the paper. The author also thanks anonymous reviewers for helpful comments that improved the presentation of the paper.

1 Introduction

If we want to catch a lion in the desert, a binary search is the method of choice: We bipartition the desert, check in which cell the lion resides, and recurse. But what if we want to catch a lion in a lions den, where lions are in abundance?

We are interested in the following problem. A deterministic algorithm tries to find a string in an unknown set $S \subseteq \{0,1\}^n$, where it is a-priori guaranteed that the density of S is large (e.g., $|S| \geq 2^{n-1}$). The only information that the algorithm can obtain about S is an *estimation* of the density of S in any subset $Q \subseteq \{0,1\}^n$ of its choice. That is, for any subset $Q \subseteq \{0,1\}^n$, the algorithm can obtain a value $\tilde{\nu}(Q)$ such that $\tilde{\nu}(Q) = \frac{|Q \cap S|}{|Q|} \pm \mu$, for a small $\mu > 0$. Can the algorithm find a string $s \in S$ more efficiently than simply going over all singletons in $\{0,1\}^n$ and checking whether or not each of them is in S?

As noted by Goldreich [1, Thm. 3.5], if the estimation error μ is sufficiently small, then the problem can be solved efficiently, using a method similar to the method of conditional probabilities. Specifically, the algorithm iteratively constructs a string $s \in S$ bit-by-bit, where in each iteration the algorithm decides which value for the next bit would yield a higher density of S in the resulting subcube, up to an error of μ. Unfortunately, this method requires that the error μ will be inversely proportional to the number of iterations (i.e., $\mu = 1/O(n)$). Moreover, the method has the drawback of being *inherently sequential*: Constructing an n-bit solution involves sequentially solving n decision problems. Thus, our main question in this work is the following:

Can the problem be solved more efficiently, compared to the naive "equipartition and recurse" algorithm? Specifically, can we improve the dependency on the estimation error, and can a parallel algorithm solve the problem faster?

The problem that we study in this work is especially relevant in the context of *derandomization*. Think of S as the set of satisfying inputs for some circuit $C : \{0,1\}^n \to \{0,1\}$. Two fundamental problems in derandomization are the problem of *hitting* the set S (i.e., finding a satisfying input for C), and the problem of *approximately counting* the size of S (i.e., estimating the acceptance probability of C; this problem is sometimes called the *Circuit Acceptance Probability Problem*). From this perspective, the question underlying the current work is the following: Does the hitting problem for a circuit C reduce to the approximate counting problem for C (on subsets of $\{0,1\}^n$), in general? [1] And more specifically, can we improve on the sequential reduction that was presented above if we are given only limited "non-black-box" information about C?

The problem that we study can also be viewed as an extension of two classical problems. Specifically, consider the problem of reducing the search for a string in a non-empty set S to the task of deciding, for any $Q \subseteq \{0,1\}^n$, whether or not $Q \cap S \neq \emptyset$ (see [6]). Our problem is a natural extension of this problem to the setting where the target set S is *dense*, with the corresponding decision task adapted accordingly (to estimating the *density* of S in any subset Q). Moreover, our problem is a variant of an open problem of Motwani, Naor and Naor [7]: They also considered a dense set S, but in their setting the algorithm can obtain the *exact* density of S in any subset $Q \subseteq \{0,1\}^n$, rather than a density estimation. Indeed,

[1] In typical settings, if one can approximate the acceptance probability of C, then one can also approximate the acceptance probability of C on subcubes of $\{0,1\}^n$. Our main lower bounds hold even if the algorithm can approximate the acceptance probability of C on *any* subset of $\{0,1\}^n$, whereas the upper bounds only rely on such estimations on subcubes.

their setting is also natural, but less relevant to derandomization than our setting. They conjectured that in their setting, even if the algorithm can obtain, in each step, the density of S in $\text{poly}(n)$ sets in parallel, still no significant speed-up over the method of conditional probabilities is possible (for details see Section 2). As far as we know, no progress has been made on their conjecture prior to this work.

1.1 Lower bounds on parallel algorithms

The first question that we consider is whether the problem described above can be solved faster by a parallel algorithm. Specifically, assume that the algorithm searching for a string in S works in *iterations*. In each iteration, the algorithm sends $p = p(n)$ queries to a *density oracle*, where each question is a set $Q_i \subseteq \{0,1\}^n$, and then receives p answers, where each answer is a density estimation $\tilde{\nu}(Q_i) = \frac{|Q_i \cap S|}{|Q_i|} \pm \mu$. Can such an algorithm find a string $s \in S$ using less than n iterations?

If μ is small, one can obtain an efficient algorithm that uses $n/\log(p)$ iterations, by a straightforward adaptation of the method of conditional probabilities: Instead of constructing a solution bit-by-bit, construct a solution block-by-block, where each block consists of $\log(p)$ bits. This yields the following upper bound.

▶ **Theorem 1.** *(an upper bound for parallel algorithms; informal). For any $p \in \mathbb{N}$ and $\mu < \frac{\log(p+1)}{4n}$, an algorithm that uses p density estimations with error μ in each iteration can find a string in an unknown set S of size $|S| \geq 2^{n-1}$ using $\frac{n}{\log(p+1)}$ iterations.* [2]

Our main result for the setting of parallel algorithms is that, unless the estimation error is extremely small (i.e., unless $\mu = o(1/\text{poly}(p))$), the algorithm described above essentially cannot be improved upon. In particular, for the natural setting of $p = \text{poly}(n)$ and $\mu = 1/\text{poly}(n)$, the problem requires an almost-linear number of iterations to solve.

▶ **Theorem 2.** *(a lower bound on the number of iterations of parallel algorithms; informal). For any $\mu > 0$ and $p \in \mathbb{N}$, algorithms that use p density estimations with error μ in each iteration need at least $\frac{n}{\log(p+1)+\log(1/\mu)}$ iterations to find a string in a set S of size $|S| \geq 2^{n-1}$.*

The lower bound in Theorem 2 is proved under the assumption that $|S| \geq 2^{n-1}$. One might expect that if S has significantly larger density (e.g., $|S| = (1 - o(1)) \cdot 2^n$), then an algorithm might be able to find a string in S using less iterations. Our second result shows that even if $|S| \geq (1 - 2^{-\Omega(n)}) \cdot 2^n$, then the lower bound asserted in Theorem 2 still essentially holds. Actually, we generalize Theorem 2, by showing a trade-off between the density of S and a lower bound on the number of iterations required to find a string in S.

▶ **Theorem 3.** *(a lower bound for parallel algorithms and large sets; informal). For any $0 < \mu \leq 1/2$, and $p \in \mathbb{N}$, and $\epsilon > 0$, algorithms as above need at least $\frac{\epsilon \cdot n}{\log(p+1)+\log(1/\mu)}$ iterations in order to find a string in an unknown set S of size $|S| \geq 2^n - 2^{\epsilon \cdot n}$.*

Recall that we are particularly interested in this problem when S is the set of satisfying inputs for some circuit $C : \{0,1\}^n \to \{0,1\}$, and the algorithm tries to find a satisfying input for C by estimating the acceptance probability of C in subsets of $\{0,1\}^n$. In this setting one does not necessarily expect the algorithm to be able to estimate the acceptance probability

[2] To obtain an upper bound of $n/\log(p+1)$, instead of $n/\log(p)$, the algorithm partitions the space in each iteration into $p+1$ sets, and relies on the estimations for the density of S in the first p sets in order to estimate the density of S in the $(p+1)^{th}$ set.

of C in very "complex" subsets. When we impose such a limitation on the circuit complexity of the queries that the algorithm makes, we obtain the following strengthening of Theorem 3, which asserts that the same lower bound holds even if the algorithm is guaranteed that S can be decided by a relatively simple circuit.

▶ **Theorem 4.** *(the lower bound for parallel algorithms holds even for "simple" circuits; informal). Assume that for any query $Q \subseteq \{0,1\}^n$ that the algorithm from Theorem 3 makes, the set Q can be decided by a circuit from a circuit class $\mathcal{C}$. Then, the lower bound in Theorems 3 holds even if the algorithm is guaranteed that the set S can be decided by a conjunction of negations of $n \cdot p$ circuits from $\mathcal{C}$.*

One corollary of Theorem 4 is that if the algorithm only estimates the acceptance probability of C on *subcubes* of $\{0,1\}^n$, then even the guarantee that C is a polynomial-sized CNF does not allow to bypass the lower bound in Theorem 3.

Note that the size of the circuit for S in Theorem 4 is larger than the total number of queries made by the algorithm. This is no coincidence: If the algorithm is given the size of C, and is allowed to make a number of queries that is polynomial in the size of C, then the algorithm can simply query, in parallel, the singletons in the output-set of a pseudorandom generator for C (assuming that such a generator exists; see, e.g., [12, Prop 7.8]). In contrast, Theorem 4 asserts that when the number of queries is *smaller* than the size of the circuit, and the algorithm uses density estimations (rather than only query singletons), the guarantee that C is a conjunction of negations of $n \cdot p$ circuits from $\mathcal{C}$ does not suffice in order to bypass the lower bound in Theorem 3.

When S is the set of satisfying inputs for a circuit C, the setting of $|S| = (1 - o(1)) \cdot 2^n$ corresponds to circuits that accept almost all of their inputs. This setting, called *quantified derandomization*, has recently been introduced by Goldreich and Wigderson (see [2, 9, 10]).

1.2 Lower bounds on algorithms with large estimation error

The second question that we consider in this paper is what happens when the estimation error μ is too large to use the method of conditional probabilities (i.e., $\mu = \omega(1/n)$). In this setting, we are not necessarily interested in parallel algorithms, but simply ask what is the *number of estimations* needed in order to find a string $s \in S$.

Similarly to the previous section, we show that a naive algorithmic approach essentially cannot be improved upon. Specifically, consider an algorithm that works in iterations; in each iteration, the algorithm equipartitions the "current" search space into $2^{4\mu \cdot n}$ sets, obtains estimates for the density of S in each of the sets, and recurses into the set in which S has the highest estimated density. Since the depth of the recursion tree is less than $1/4\mu$, an estimation error of μ suffices for this algorithm. [3] This yields the following upper bound.

▶ **Theorem 5.** *(an upper bound for algorithms with large estimation errors; informal). For any error $\mu > 0$, an algorithm that only uses density estimations with error μ can find a string in an unknown set S of size $|S| \geq 2^{n-1}$ using less than $\frac{2}{\mu} \cdot 2^{4\mu \cdot n}$ density estimations.*

Note that when the estimation error is $\mu = O(\log(n)/n)$, the algorithm described above uses poly(n) density estimations. Our main result in the current section is that whenever $\mu \geq \frac{4 \cdot \log(n)}{n}$, the upper bound in Theorem 5 is essentially tight. To see this, observe that

[3] Indeed, when $\mu = O(\log(n)/n)$, this is essentially the same algorithm as the parallel algorithm from Theorem 1; the number of estimations in each iteration is $p = 2^{4 \cdot \mu \cdot n} = \text{poly}(n)$.

when $\mu \geq \frac{4 \cdot \log(n)}{n}$, the upper bound in Theorem 5 is $\frac{2}{\mu} \cdot 2^{4\mu \cdot n} = 2^{O(\mu \cdot n)}$; and when $\mu > 1/4$, the upper bound exceeds 2^n. Thus, it is nearly matched by the following lower bound.

▶ **Theorem 6.** *(a lower bound on the number of estimations needed by algorithms with large estimation errors; informal). For any error $\mu \geq \frac{4 \cdot \log(n)}{n}$, at least $2^{\Omega(\mu \cdot n)}$ density estimations are needed in order to find a string in an unknown set S of size $|S| \geq 2^{n-1}$. Moreover, if $\mu \geq \frac{1}{4} + \Omega(1)$, then $\Omega(2^n/n)$ estimations are needed to find a string in S of size $|S| \geq 2^{n-1}$.*

Theorem 6 implies in particular that if the error satisfies $\mu = \omega(\log(n)/n)$, then the problem cannot be solved efficiently (i.e., using only $\text{poly}(n)$ estimations). We also generalize Theorems 5 and 6, by showing a trade-off between the density of S, denoted by ρ, and the number of estimations required to find a string in S. This generalization is most interesting when considering sets S with small density (e.g., density $\rho = O(\mu)$), in which case the problem is much more difficult.

▶ **Theorem 7.** *(a general trade-off between density, error, and the number of queries needed to solve the problem; informal). For any error $\mu \geq 12 \cdot \log(n)/n$, the following holds:*

1. *For any density $\rho \leq (2 - \Omega(1)) \cdot \mu$, at least $2^n/\text{poly}(n)$ density estimations are needed to find a string in a set S of size $|S| \geq \rho \cdot 2^n$.*
2. *For any density $\rho \geq (2 - o(1)) \cdot \mu$, it holds that $2^{\Theta((\mu/\rho) \cdot n)}$ density estimations are necessary and sufficient to find a string in a set S of size $|S| \geq \rho \cdot 2^n$.*

1.3 Lower bounds on algorithms with *no* estimation error

Finally, we consider the setting suggested by Motwani, Naor, and Naor [7], in which there is no estimation error (i.e., $\mu = 0$). They conjectured that a naive "equipartition and recurse" will still be essentially optimal in this setting.

We focus on the question of the required number of queries to solve the problem (and leave open the question of parallelism). Recall that the algorithm obtains density values, and not binary answers, and thus a naive information-theoretic lower bound of n queries does not hold. Nevertheless, we show that $n - O(1)$ queries are still necessary.

▶ **Theorem 8.** *(the minimal number of* exact *density queries; informal). Consider algorithms that, for an unknown set S of size $|S| > \rho \cdot 2^{n-1}$, can query an oracle to obtain the exact density of S in any subset $Q \subseteq \{0,1\}^n$. Then, the number of queries that such algorithms need in order to find a string $s \in S$ is at least $n - \lfloor \log(1/(1-\rho)) \rfloor - 1$.*

The lower bound in Theorem 8 is tight, up to a single bit. The proofs of Theorem 8 and of the corresponding upper bound appear in [8, Sec. 7].

1.4 Organization

In Section 2 we discuss the context of our results and previous related work. In Section 3 we explain the techniques used to obtain our results, in high-level. Section 4 contains the formal definitions of the algorithms described above. In Section 5 we prove the lower bounds on solving the problem in parallel (i.e., Theorems 2 and 3), and the corresponding upper bound (i.e., Theorem 1) is proved in [8, Apdx. A]. In Section 6 we prove the lower bounds on solving the problem when the estimation error is large (i.e., Theorems 6 and 7), and the corresponding upper bounds (i.e., Theorem 5 and the upper bound in Item (2) of Theorem 7) are proved in [8, Apdx. A].

2 Background and previous works

Several years ago, Goldreich [1] considered the hypothesis that *promise-$\mathcal{BPP}$ = promise-$\mathcal{P}$*, which in particular implies that the density of satisfying inputs for a given circuit can be estimated in polynomial time. Goldreich showed that it follows that *$\mathcal{BPP}$ search problems* (see the definition in [1, Sec. 3]) can also be solved in deterministic polynomial time, and in particular, there exists a deterministic polynomial-time algorithm that finds a satisfying input for a given circuit that accepts most of its inputs. Indeed, his reduction of search problems to problems of estimating the density of solutions in subsets of the search space is based on the method of conditional probabilities. The current work can be viewed as an extension of his study, which considers a more generic reduction of the task of finding a satisfying input for a circuit C to estimating the density of satisfying inputs for C in subsets of the domain, [4] and asks whether his solution for this problem can be improved upon, in general, with only limited "non-black-box" information about the circuit C.

Thus, this work is situated within a line of works that study the limitations of "black-box" techniques in derandomization. Although many of the best current derandomization results are based on constructions that are essentially black-box (i.e., on pseudorandom generators that use very little information about the circuit that they wish to "fool"), black-box techniques nevertheless have disadvantages in the context of derandomization. In particular, constructing a black-box pseudorandom generator (or a hitting-set generator) for a circuit class necessitates proving strong corresponding lower bounds against that circuit class; [5] and black-box techniques cannot be used in certain settings for hardness amplification, which is a common strategy to try and prove the lower bounds necessary to construct pseudorandom generators via the hardness-randomness paradigm (see, e.g., [13, 11]).

The current work also extends the study of Karp, Upfal, and Wigderson [6], who proved lower bounds for the setting in which the target set S is only guaranteed to be non-empty (rather than dense), and the algorithm can obtain, for any set $Q \subseteq \{0,1\}^n$, an answer to whether or not a solution exists in Q (i.e., whether or not $Q \cap S = \emptyset$). The authors showed that the "equipartition-and-recurse" strategy is optimal in this setting, since any strategy requires $n/\log(p+1)$ iterations to find a solution, in general, where p is the number of decision problems that can be solved in parallel in each iteration.

Motwani, Naor, and Naor [7] considered a setting in which S can be dense (rather than just non-empty), and conjectured that a similar lower bound would hold in this setting even if the algorithm trying to find a string $s \in S$ can obtain, for any $Q \subseteq \{0,1\}^n$, the exact number of solutions in Q (i.e., the value $|S \cap Q|$). Thus, Theorems 2 and 3 affirm a weak version of their conjecture, where the difference is that in our case, instead of obtaining the exact number of solutions in Q, the algorithm can only obtain an estimate of the number of solutions in Q. In addition, Theorem 8 proves their conjecture for the special case of $p = 1$ (i.e., when there is no parallelism).

3 Our techniques

To understand the challenge, let us first recall the techniques of Karp, Upfal, and Wigderson [6]. They proved their lower bound (for algorithms that can only probe for the existence of a

[4] That is, we only consider the hypothesis that one can efficiently estimate the density of satisfying inputs for C on subsets of its domain, rather than the hypothesis that *promise-$\mathcal{BPP}$ = promise-$\mathcal{P}$*.

[5] Analogous implications of *any* derandomization of a circuit class (i.e., not necessarily a black-box one) are known, but the implied lower bounds are weaker and less direct (see, e.g., [3, 4, 5, 14]).

solution in any subset) by an adversarial argument: For any algorithm A, they simulated the execution of A, and supplied adversarial answers, in order to delay A's progress in finding $s \in S$. In their argument, in each iteration, the adversary has to answer A's queries in a manner that is consistent with the current information available to A (i.e., with all previous answers). However, since the adversary only provides "yes/no" answers, relatively little information is revealed about S in each iteration, and thus relatively few constraints are imposed upon the adversary when engineering answers in subsequent iterations.

As noted by [7], it is not a-priori clear how to extend the foregoing strategy to a setting in which A obtains the *exact* number of solutions in any subset that it queries. This is the case because in the latter setting, the adversary has to answer A's queries with exact density values that are *perfectly consistent* with some fixed set S; this requirement imposes strict constraints on the adversary in each iteration. This seems to require much more careful engineering of answers on the adversary's part.

The key observation underlying our lower bounds is that if we, as adversaries, are allowed a small error in our answers to A, then we do not need to engineer the answers so carefully. One approach to take advantage of this relaxed setting, which we use in the proofs of Theorems 2 and 3, is to start the simulation with a "tentative" set S, modify this set adversarially throughout the execution, and answer A in each iteration according to the current state of S (instead of engineering artificial answers). If the modifications that we make to the set S throughout the execution are not too substantial, then the final version of S is not very different from any of the tentative ones, which implies that the error in our answers was never too big. This approach also allows us to show that there exists a relatively simple circuit that decides S (i.e., to obtain Theorem 4).

An alternative approach, which we use in the proofs of Theorems 6 and 7, is to answer A's queries according to some set of rules, and in the end construct *an adversarial distribution* of sets such that a set sampled from the distribution will be consistent with our answers, up to a small error, with high probability. That is, the fact that we are allowed a small error allows us to avoid the explicit construction of a single adversarial set, and instead rely on an adversarial distribution of sets. Specifically, we will answer each query $Q \subseteq \{0,1\}^n$ of A only according to the size of Q; and in the end we will construct a distribution $\mathcal{S}_A$ over sets $S \subseteq \{0,1\}^n$, which depends on the specific queries that A issued, such that a set $S \sim \mathcal{S}_A$ will be consistent with our answers, up to an error of μ, with high probability.

In contrast, in Theorem 8 we consider algorithms without an estimation error, and thus we indeed need to fully engineer exact adversarial answers. To do so, we maintain a template for the set S, which is a partition of $\{0,1\}^n$ such that each set P in the partition is labeled with the density of S in P. Given each query of A, we refine the partition, while making sure that unless the partition is extremely refined, no set P in the partition is fully contained in S. Thus, the algorithm needs to use many queries, in order to yield an extremely refined partition, which will allow it to find a singleton $s \in S$.

4 Preliminaries

All logarithms in the paper are to base 2. We formally define the algorithms described in Section 1 by using the notion of oracle machines, where the oracle is the device supplying density estimations. An oracle function for a set S gets as input a sequence of p density queries, and outputs p estimations for the density of S in each of the queried sets, where each estimation is correct up to a relative additive error of μ.

▶ **Definition 9.** (p-parallel μ-error density estimators). For $n, p \in \mathbb{N}$, and $\mu < 1$, and a set $S \subseteq \{0,1\}^n$, a function $f_S : \mathcal{P}(\{0,1\}^n)^p \to [0,1]^p$ is called a p-parallel μ-error density estimator for S if for every $\vec{Q} = (Q_1, Q_2, ..., Q_p) \in \mathcal{P}(\{0,1\}^n)^p$, and every $j \in [p]$, it holds that $\left|\tilde{\nu}(Q_j) - \frac{|Q_j \cap S|}{|Q_j|}\right| \le \mu$, where $\tilde{\nu}(Q_j)$ is the j^{th} element in the sequence $f_S(\vec{Q})$. [6]

▶ **Definition 10.** (hitters with access to density estimators). Let $\mu : \mathbb{N} \to [0,1)$, and let $p : \mathbb{N} \to \mathbb{N}$. A deterministic algorithm A is called a hitter with oracle access to p-parallel μ-error density estimators if for every $n \in \mathbb{N}$ and $S \subseteq \{0,1\}^n$, given input 1^n and oracle access to a $p(n)$-parallel $\mu(n)$-error density estimator for S, the algorithm A outputs a string $s \in S$.

We deliberately avoid the question of how A specifies its queries to the oracle, and just assume that all queries can be perfectly communicated. Our lower bounds are thus solely information-theoretic. On the other hand, the algorithms establishing the upper bounds in the paper only use queries about subcubes of $\{0,1\}^n$, which can be easily communicated in any reasonable model of an oracle Turing machine.

When $p = 1$ (i.e., when there is no parallelism), we just refer to a `$\mu$-error density estimator` and to a `hitter with oracle access to $\mu$-error density estimators`. A hitter as in Definition 10 operates in iterations, where in each iteration it issues a query-tuple to the oracle (i.e., a sequence of p sets), and receives an answer-tuple (i.e., p correponsing density estimations). When we will discuss a specific query (resp., specific answer), we will usually mean one of the p sets queried in some iteration (resp., one of the p density estimations given in the answer).

5 Lower bounds on parallel algorithms

Let us begin by stating Theorem 2 and proving it. For simplicity, we will prove a lower bound of $\frac{n}{\log(p)+\log(1/\mu)+1}$, instead of $\frac{n}{\log(p+1)+\log(1/\mu)}$. [7]

▶ **Theorem 11.** *(a lower bound for parallel algorithms; Theorem 2, restated). For $\mu : \mathbb{N} \to (0, \frac{1}{2})$ and $p : \mathbb{N} \to \mathbb{N}$, let A be a* hitter with oracle access to p-parallel μ-error density estimators. *Then, for any $n \in \mathbb{N}$, there exists a set $S \subseteq \{0,1\}^n$ of size $|S| \ge 2^{n-1}$ and a $p(n)$-parallel $\mu(n)$-error density estimator f_S for S such that the number of iterations that A uses when given oracle access to f_S is at least*

$$\frac{n}{\log(p(n)) + \log(1/\mu(n)) + 1} .$$

Proof overview. Let $n \in \mathbb{N}$, and let $p = p(n)$ and $\mu = \mu(n)$. Assume towards a contradiction that A always uses $R < \frac{n}{\log(p)+\log(1/\mu)+1}$ iterations. We will construct a set S and a p-parallel μ-error density estimator f_S that "fool" A: That is, f_S answers all of A's queries in a manner that is consistent with S, up to a relative error of μ, but in the end of the execution of A, the algorithm outputs a string that is not in S.

It will be more convenient to work with a definition for hitters that is slightly different from the one in Definition 10: Instead of requiring that the algorithm outputs a string $s \in S$ in the end of the execution, we require that the hitter will ask the oracle about a singleton $Q = \{s\}$, and receive an answer $\tilde{\nu}(\{s\}) > \mu$ (which implies that $s \in S$). The number of iterations required to solve the problem is identical in both definitions, up to ± 1 iteration.

We will simulate the execution of A, and answer the algorithm's queries adversarially, in order to delay its progress in finding small sets that contain elements in S. Specifically,

[6] We denote by $\mathcal{P}(\{0,1\}^n)$ the power set of $\{0,1\}^n$.
[7] The proof of the slightly tighter lower bound appears in [8, Thm. 11].

let us call a set Q a *positive set* if at some point during the execution of A, the algorithm queried the oracle about the set Q and was answered by a non-zero value (i.e., by $\tilde{\nu}(Q) > 0$). Our goal is that in the end of the execution, all positive sets will be of size at least 2, which will imply that A did not find any positive singleton.

Our answers throughout the execution of A have to be consistent with some fixed set S, up to an estimation error of μ. To ensure this, we will maintain a "tentative" version of the set S, and provide answers in each iteration according to the tentative set at that iteration. We will show that the final version of the set S, which is the tentative set in the end of the execution, is not very different from any of the non-final versions. Thus, the answers given to A are consistent, up to a small error (less than μ), with the set S.

We initialize the tentative set as $S_0 = \{0,1\}^n$. In iteration $i \in [R]$, we modify the current tentative set, denoted S_{i-1}, and obtain the new tentative set, S_i, as follows:

- We start iteration i with a guarantee that all positive sets are of size at least h_i (where h_i is a parameter to be determined).
- Given A's queries, we remove from the tentative set all the strings from queried-sets that are "too small". That is, the new tentative set S_i is obtained by removing from S_{i-1} every string that belongs to a queried set Q' such that $|Q'| < \ell_i$ (where ℓ_i is also a parameter to be determined).
- We answer the queries of A according to the (current) tentative set S_i; that is, the query Q is answered by $|Q \cap S_i|/|Q|$. Thus, in the next iteration, all positive sets will be of size at least $h_{i+1} = \ell_i$.

We define S to equal the tentative set at the end of the execution (i.e., $S = S_R$). Let us now describe our setting of parameters, and explain why it suffices to prove the theorem. In the first iteration we have $h_1 = 2^n$, which holds vacuously. We define ℓ_i in each iteration such that $\ell_i/h_i = \mu/(2 \cdot p)$. It follows that $\ell_R = (\mu/(2p))^R \cdot h_1 = (2p/\mu)^{-R} \cdot 2^n$. Relying on the hypothesis that $R < \frac{n}{\log(p)+\log(1/\mu)+1} = \log_{2p/\mu}(2^n)$, we have that $\ell_R > 1$, which means that A did not find any positive singleton.

Now, let Q be a positive set that was queried in iteration i, and let us count the number of strings removed from the tentative set in subsequent iterations. The number of strings removed in iteration $i+1$ is less than $p \cdot \ell_{i+1} = (\mu/2) \cdot h_{i+1} = (\mu/2) \cdot \ell_i$. Since in iteration i we answer positively only for sets of cardinality at least ℓ_i, we know that $|Q| \geq \ell_i$, and thus the number of strings removed in iteration $i+1$ is less than $(\mu/2) \cdot |Q|$. In subsequent iterations, the number of strings removed from the tentative set decays exponentially (see [8, Claim 11.4]); hence, when summing over iterations $i+1, ..., R$, the overall number of strings removed is less than $\mu \cdot |Q|$. Similarly, the overall number of strings removed from $S_0 = \{0,1\}^n$ is less than $\mu \cdot |S_0|$, and thus we have that $|S| > 2^{n-1}$. For full details see [8, Thm. 11]. ◀

We now prove Theorem 3, which asserts a trade-off between the density of S and a lower bound on the number of iterations needed to find a string $s \in S$.

▶ **Theorem 12.** *(a lower bound for parallel algorithms and large sets; Theorem 3, restated). For $\mu : \mathbb{N} \to (0, \frac{1}{2})$ and $p : \mathbb{N} \to \mathbb{N}$, let A be a hitter with oracle access to p-parallel μ-error density estimators. Then, for any $\epsilon > 0$ and $n \in \mathbb{N}$, there exists* a set $S \subseteq \{0,1\}^n$ of size *$|S| \geq 2^n - 2^{\epsilon \cdot n}$ and a $p(n)$-parallel $\mu(n)$-error density estimator f_S for S such that the number of iterations that A uses when given oracle access to f_S is at least $\frac{\epsilon \cdot n}{\log(p(n))+\log(1/\mu(n))+1}$.* [8]

[8] For proof of the slightly stronger lower bound $\frac{\epsilon \cdot n}{\log(p+1)+\log(1/\mu)}$ see [8, Thm. 12].

Proof overview. In the proof of Theorem 11, the threshold that defines a "small" query starts out quite high; that is, $\ell_1 = \alpha \cdot 2^n$, where $\alpha = (\mu/2p)$. (In each subsequent iteration, the threshold decreases by a multiplicative factor of α.) Thus, in the first iteration we might remove $\ell_1 \cdot p \approx \mu \cdot 2^n$ strings from the tentative set.

In the current proof we wish to avoid the removal of so many strings from the tentative set. To do so, we will set the threshold *in the first iteration* to be about $2^{\epsilon \cdot n}$. Specifically, denoting by $\mathcal{R} = \frac{n}{\log(p)+\log(1/\mu)+1}$ the number of iterations in the proof of Theorem 2, we will set the threshold ℓ_1 to the value that it obtained (in the proof of Theorem 2) in iteration $i \approx (1-\epsilon) \cdot \mathcal{R}$; that is, $\ell_1 \approx \alpha^{(1-\epsilon)\cdot \mathcal{R}} \cdot 2^n = 2^{\epsilon \cdot n}$. Then, in each subsequent iteration, we will decrease the threshold by a multiplicative factor of α. The number of removed strings will thus be less than $2^{\epsilon \cdot n}$, whereas the number of iterations (i.e., the lower bound) will be $\epsilon \cdot \mathcal{R} = \frac{\epsilon \cdot n}{\log(p)+\log(1/\mu)+1}$. For full proof details, see [8, Thm. 12]. ◀

The circuit complexity of the "hard" set S.

Recall that a motivating example for the problem discussed in this paper is when S is the set of satisfying inputs for some circuit C, and the algorithm A tries to find a satisfying input for C. One might intuitively expect that in order to construct a "hard" set S for A (i.e., a set S that forces A to use many iterations), a "complicated" circuit C will be needed. However, we observe that the circuit complexity of the "hard" sets that are constructed in the proofs of Theorems 11 and 12 is proportional only to the circuit complexity of the sets that A queried. Thus, we can now prove Theorem 4, which asserts that the lower bound in Theorem 12 holds even if the algorithm is guaranteed that S can be decided by a relatively simple circuit:

▶ **Theorem 13.** *(the lower bound on parallel algorithm holds even for "simple" sets; Theorem 4, restated). Let μ, p, and A be as in Theorem 12, and further assume that for every query $Q \subseteq \{0,1\}^n$ that A makes, the set Q can be decided by a circuit from a circuit class $\mathcal{C}$. Then, for any $\epsilon > 0$ and $n \in \mathbb{N}$, there exists* a set $S \subseteq \{0,1\}^n$ of size $|S| \geq 2^n - 2^{\epsilon \cdot n}$ that can be decided by an conjunction of negations of at most $n \cdot p$ circuits from $\mathcal{C}$ *and a $p(n)$-parallel $\mu(n)$-error density estimator f_S for S such that the number of iterations that A uses when given oracle access to f_S is at least $\frac{\epsilon \cdot n}{\log(p(n))+\log(1/\mu(n))+1}$.*[9]

Proof. The set S that is constructed in the proof of Theorem 12 is obtained by starting from the tentative set $S_0 = \{0,1\}^n$, and removing subsets that correspond to some of the algorithm's queries (i.e., the ones that were deemed "too small" in the relevant iteration). Thus, the set S is the intersection of the complements of at most $(\epsilon \cdot \mathcal{R}) \cdot p < n \cdot p$ subsets such that each subset can be decided by a circuit from $\mathcal{C}$. ◀

6 Lower bounds on algorithms with large estimation error

We now formally state Theorem 6, and prove the first part of the theorem; for the "moreover" part, see [8, Prop. 14]. In fact, we will prove a statement slightly stronger than the one in the first part of Theorem 6: Instead of proving the lower bound when the set S of size $|S| \geq 2^{n-1}$, we will prove it assuming that $|S| \geq (1-\mu) \cdot 2^n$.

▶ **Theorem 14.** *(a lower bound for large errors; Theorem 6, restated). Let $\mu : \mathbb{N} \to (0, 1/2)$ such that $\mu(n) \geq \frac{4 \cdot \log(n)}{n}$, and let A be a hitter with oracle access to μ-error density estimators.*

[9] For proof of the slightly stronger lower bound $\frac{\epsilon \cdot n}{\log(p+1)+\log(1/\mu)}$ see [8, Thm. 13].

Then, for any sufficiently large $n \in \mathbb{N}$, there exists a set $S \subseteq \{0,1\}^n$ of size $|S| \geq (1-\mu)\cdot 2^n$, and a μ-error density estimator f_S for S, such that the number of iterations that A uses when given oracle access to f_S is at least $2^{\Omega(\mu \cdot n)}$.

Proof overview. Let $\mu' = \mu/2$, and for simplicity, assume that μ is constant. Similar to the proofs in Section 5, we simulate A and provide adversarial answers. However, in the current proof our answers will be given according to one fixed rule, which does not change throughout the execution (in contrast to the previous proofs, in which the "threshold" for defining small sets decreased in each iteration).

The main idea is to arrange all sets of size more than n in "levels", where the i^{th} level contains sets of size between $2^{(i-1)\cdot\mu'\cdot n}$ and $2^{i\cdot\mu'\cdot n}$. Then, during the execution, we will output the same fixed estimate for all queried-sets in the same level; specifically, for sets in level i, we will output the estimate $i\cdot\mu'$. Note that if we use $1/\mu'$ levels, then our estimate for the set $\{0,1\}^n$, which is in the highest level, is $\tilde{\nu}(\{0,1\}^n) = 1$; this implies that any S consistent with our answers has density at least $1-\mu$. To see that there exists a set S that is consistent with such answers, fix a queried-set Q in level i. If the algorithm makes at most $R \ll 2^{\mu'\cdot n}$ queries, then the vast majority of strings in Q do not belong to queried-sets of level $i-2$ or less, since the number of such strings is at most $R\cdot 2^{(i-2)\cdot\mu'\cdot n} \ll |Q|$. Thus, if we include every string $w \in Q$ in S with probability $\ell(w)\cdot\mu'$, where $\ell(w)$ is the level of the smallest queried-set that contains w, then the vast majority of strings in Q will be included in S with probability either $(i-1)\cdot\mu'$ or $i\cdot\mu'$. Hence, the expected density of S in Q will be close to the interval $\big[(i-1)\cdot\mu', i\cdot\mu'\big]$, which implies that, with high probability, the actual density of S in Q will not deviate from our estimate of $\tilde{\nu}(Q) = i\cdot\mu'$ by more than $2\cdot\mu' = \mu$.

Let us now provide further details for this idea. We partition the power set of $\{0,1\}^n$ into levels as follows. The zero level, denoted $\mathcal{L}_0$, consists of all sets of size at most (roughly) n. For $i = 1,...,1/\mu'$, the i^{th} level, denoted $\mathcal{L}_i$, consists of all sets that are not included in any level $j < i$, and that are of size at most $2^{i\cdot\mu'\cdot n}$. Observe that for every $i \geq 3$, every set in $\mathcal{L}_i$ is larger than every set in $\mathcal{L}_{i-2}$ by a multiplicative factor of at least $2^{\mu'\cdot n}$. After $R \approx \mu'\cdot 2^{\mu'\cdot n}$ iterations, in which we act as above (i.e., answer $\tilde{\nu}(Q) = i\cdot\mu'$ for $Q \in \mathcal{L}_i$), we let S be a random set such that every string $w \in \{0,1\}^n$ is included in S, independently, with probability $\ell(w)\cdot\mu'$ (recall that $\ell(w)$ is the level of the smallest queried-set that contains w, or $\ell(w) = 1/\mu'$, if w was not included in any queried-set).

Note that A cannot find a positive singleton, since we output the estimate zero for every queried-set in $\mathcal{L}_0$ (i.e., every set of size smaller than (roughly) n). Also note that strings in queried-sets in $\mathcal{L}_0$ are never included in S, and thus our estimate (of zero) for every queried-set in $\mathcal{L}_0$ is always correct. Our main claim is that, with high probability, *for any queried-set $Q \in \mathcal{L}_i$, where $i \geq 1$, the density of S in Q, denoted by $\delta_S(Q)$, satisfies $i\cdot\mu' - \mu \leq \delta_S(Q) \leq i\cdot\mu' + \mu$.* This claim implies that our estimate for Q is correct, up to an error of μ. Let us sketch the proof of this claim.

- To see that $\delta_S(Q) \leq i\cdot\mu' + \mu = (i+2)\cdot\mu'$, note that every $w \in Q$ is included in S with probability $\ell(w)\cdot\mu' \leq i\cdot\mu'$, where the inequality is because $\ell(w)$ is upper bounded by the level of Q, which is i.
- To see that $\delta_S(Q) \geq i\cdot\mu' - \mu = (i-2)\cdot\mu'$, first note that this lower bound is trivial for $i \leq 2$ (because $i-2 \leq 0$). If $i \geq 3$, the number of strings from queried-sets of level $j \leq i-2$ in Q is at most $R\cdot\max_{Q'\in\mathcal{L}_{i-2}}|Q'|$. Now, recall that every set in $\mathcal{L}_i$ is larger than every set in $\mathcal{L}_{i-2}$ by a multiplicative factor of at least $2^{\mu'\cdot n}$, and that $R \approx \mu'\cdot 2^{\mu'\cdot n}$. Hence, the vast majority of strings $w \in Q$ satisfy $\ell(w) \geq i-1$, and they will be included in S with probability at least $(i-1)\cdot\mu'$.

The foregoing establishes that a random set is consistent with our answers to A. Since we output the estimate 1 for any set in level $i = 1/\mu'$, and in particular for the set $\{0,1\}^n \in \mathcal{L}_{1/\mu'}$, it follows that the overall density of S is at least $(1-\mu)\cdot 2^n$. For full proof details, which also handle a sub-constant error μ, see [8, Thm. 15]. ◀

We now prove the lower bound in Item (2) of Theorem 7, which generalizes Theorem 6, by asserting a trade-off between the density of S and the number of estimations required to find a string in it. The proof of Item (1) of Theorem 7 appears in [8, Prop. 14].

▶ **Theorem 15.** *(a lower bound for large errors and arbitrary density; Theorem 7, restated). Let $\mu : \mathbb{N} \to (0,1/2)$ such that $\mu(n) \geq \frac{12\cdot\log(n)}{n}$, and let A be a hitter with oracle access to μ-error density estimators. Then, for any sufficiently large $n \in \mathbb{N}$, and any density $\rho \in [\mu(n), 1-\mu(n)]$, there exists a set $S \subseteq \{0,1\}^n$ of size $|S| \geq \rho\cdot 2^n$, and a μ-error density estimator f_S for S, such that the number of iterations that A uses when given oracle access to f_S is at least $2^{\Omega((\mu/\rho)\cdot n)}$.*

Proof overview. The proof is obtained by modifying the proof of Theorem 6. In the proof of Theorem 6, we partitioned the collection of sets of size larger than (roughly) n into $1/\mu'$ levels. The density of the set S was proportional to the *number of levels*, because we supplied the density estimate $i\cdot\mu'$ for sets in level i (and in particular, the estimate 1 for the set $\{0,1\}^n$ in level $i = 1/\mu'$). On the other hand, the lower bound on the number of estimates was proportional to the *height of each level* $i = 2,3,...,1/\mu'$, where the *height* of a level is ratio between the size of the largest set in it and the size of the smallest set in it.

In the current proof, we wish to obtain a set S with smaller density (i.e., density ρ instead of density $1-\mu$), but improve the lower bound on the number of estimates. We will do so by partitioning the subsets of $\{0,1\}^n$ into fewer levels, each of larger height. Specifically, we let $\mu' = \mu/2$, and partition the collection of sets of size more than n into slightly more than ρ/μ' levels, each of height about $2^{n/(\rho/\mu')}$. Using an analysis very similar to that in the proof of Theorem 6, we will obtain a lower bound of about $2^{(\mu'/\rho)\cdot n}$ estimates, and the set S will be of density about $(\rho/\mu')\cdot\mu' = \rho$. For full proof details, see [8, Thm. 16]. ◀

References

1 Oded Goldreich. In a world of p=bpp. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 191–232. Springer, 2011. doi:10.1007/978-3-642-22670-0_20.

2 Oded Goldreich and Avi Wigderson. On derandomizing algorithms that err extremely rarely. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 109–118, 2014. doi:10.1145/2591796.2591808.

3 R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proc. 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 734–, 1998.

4 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

5 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

6 Richard M. Karp, Eli Upfal, and Avi Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36(2):225–253, 1988.

7 Rajeev Motwani, Joseph Naor, and Moni Naor. The probabilistic method yields deterministic parallel algorithms. *Journal of Computer and System Sciences*, 49(3):478–516, 1994.

8 Roei Tell. Lower bounds on black-box reductions of hitting to density estimation. *Electronic Colloquium on Computational Complexity: ECCC*, 23:50, 2016.

9 Roei Tell. Improved bounds for quantified derandomization of constant-depth circuits and polynomials. In *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*, pages 18:1–18:49, 2017.

10 Roei Tell. Quantified derandomization of linear threshold circuits. *Electronic Colloquium on Computational Complexity: ECCC*, 24:145, 2017.

11 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

12 Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

13 Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005.

14 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal of Computing*, 42(3):1218–1244, 2013.