Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 50 (2016)

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

Roei Tell [*]

December 16, 2017

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.

[*]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Email: `roei.tell@weizmann.ac.il`

ISSN 1433-8092

Contents

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 [Gol11, 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 $Q \cap S \neq \emptyset$

[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.

(see [KUW88]). 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 [MNN94]: 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, 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]

For completeness, we provide the (straightforward) details for the proof of Theorem 1 in Appendix A.1. 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

[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.

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 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., [Vad12, 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 [GW14, Tel17a, Tel17b]).

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.*

For completeness, we provide the (straightforward) details for the proof of Theorem 5 in Appendix A.2. Note that when the estimation error is $\mu = O(\log(n)/n)$, the algorithm described above uses $\mathrm{poly}(n)$ 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 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 $\mathrm{poly}(n)$ 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 $2^n/\mathrm{poly}(n)$ 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 $|S| \geq \rho \cdot 2^n$.*

[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 $p = 2^{4 \cdot \mu \cdot n} = \mathrm{poly}(n)$.

1.3 Lower bounds on algorithms with *no* estimation error

Finally, we consider the setting suggested by Motwani, Naor, and Naor [MNN94], 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, while leaving open the question of parallelism (see Section 8 for a formulation of the latter). 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 $|S| \geq 2^{n-1}$, 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 $n-2$. Moreover, for any constant $\rho \in (0,1)$, if the size of S is $|S| = \rho \cdot 2^n$, then the number of queries required is $n - O(1)$.*

In Section 7 we state and prove a more general version of Theorem 8, which exhibits a tradeoff between the density of S and the number of exact density queries required to solve the problem.

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 Appendix A.1. 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 Appendix A.2. In Section 7 we prove the lower bound on the number of estimations needed to solve the problem when there is no estimation error (i.e., we prove Theorem 8). Finally, in Section 8 we pose a strong version of the conjecture of [MNN94] as an open question.

2 Background and previous works

Several years ago, Goldreich [Gol11] 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 [Gol11, 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., [Vio05, TV07]).

The current work also extends the study of Karp, Upfal, and Wigderson [KUW88], 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 [MNN94] 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 [KUW88]. They proved their lower bound (for algorithms that can only probe for the existence of a solution in any subset) by an adversarial argument: For

[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., [IW98, IKW02, KI04, Wil13]).

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 [MNN94], 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 $\left|\tilde{v}(Q_j) - \frac{|Q_j \cap S|}{|Q_j|}\right| \leq \mu$, where $\tilde{v}(Q_j)$ 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 μ-error density estimator and to a hitter with oracle access to μ-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

In this section we show lower bounds on solving the problem discussed in this paper "in parallel"; that is, we lower bound the number of iterations used by hitters with oracle access to a p-parallel μ-error density estimator.

5.1 The main lower bound

Let us state Theorem 2 formally, using the definitions from Section 4, and prove it.

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

[6] We denote by $\mathcal{P}(\{0,1\}^n)$ the power set of $\{0,1\}^n$.

estimators. *Then, for any $n \in \mathbb{N}$, there exists a set $S \subseteq \{0,1\}^n$ of size $|S| \geq 2^{n-1}$ 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*

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

Proof. Let $n \in \mathbb{N}$, and let $p = p(n)$ and $\mu = \mu(n)$. Assuming towards a contradiction that A always uses $R < \frac{n}{\log(p+1)+\log(1/\mu)}$ iterations, we will construct a set S and a p-parallel μ-error density estimator f_S for 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.

In the proof it will be technically 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 (as in Definition 10), we require that the hitter will ask the oracle about a singleton $Q = \{s\}$, and receive an answer $\tilde{v}(\{s\}) > \mu$ (which implies that $s \in S$). The number of iterations required to solve the problem is identical in both definitions, up to ± 1 iteration.[7]

High-level overview. For simplicity, in the overview we prove the (slightly) weaker lower bound $R \geq \frac{n}{\log(p)+\log(1/\mu)+1}$. Our strategy is similar to the one in the proof of [KUW88, Thm. 1]: We 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, 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 $\tilde{v}(Q) > 0$). 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.

Of course, 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 set" (i.e., 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

[7] This is the case because if an algorithm found a string $s \in S$, it can use an additional iteration to query $Q = \{s\}$ and obtain an estimation that is at least $1 - \mu > \mu$ (since $\mu < 1/2$); and given an answer greater than μ to a singleton query $Q' = \{s'\}$, it follows that $s' \in S$, and thus the algorithm can output s'.

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 after R iterations.

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. Observe that 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 Claim 11.4 we show that in subsequent iterations, the number of strings removed from the tentative set decays exponentially. Hence, when summing over all iterations $i+1, ..., R$, the overall number of strings removed is less than $\mu \cdot |Q|$. In similar fashion, we also show that the overall number of strings removed from $S_0 = \{0,1\}^n$ is less than $\mu \cdot |S_0|$, and thus we have that $|S| > 2^{n-1}$.

The proof details. Assume towards a contradiction that $R < \frac{n}{\log((p+\mu)/\mu)}$. Without loss of generality, assume that A never repeats the same query more than once. We construct the set S by the following procedure.

1. Let $S_0 \leftarrow \{0,1\}^n$, and let $\alpha = \mu/(p+\mu)$.
2. Run the algorithm A. For $i \in [R]$, answer the queries of A in iteration i as follows:
    (a) Let T be a temporary set, initialized with the value $T \leftarrow S_{i-1}$. Let $\vec{Q} = \left(Q_1^{(i)}, ..., Q_p^{(i)}\right)$ be the query-tuple sent by A to the oracle in iteration i.
    (b) For every $j \in [p]$ such that $|Q_j^{(i)}| < \alpha^i \cdot 2^n$, remove the strings in $Q_j^{(i)}$ from T; that is, let $T \leftarrow T \setminus Q_j^{(i)}$.
    (c) Let $S_i \leftarrow T$. Send the algorithm A the answer-tuple $\left(\tilde{v}(Q_1^{(i)}), ..., \tilde{v}(Q_p^{(i)})\right)$ such that for every $j \in [p]$ it holds that $\tilde{v}(Q_j^{(i)}) = |Q_j^{(i)} \cap S_i|/|Q_j^{(i)}|$; that is, answer according to the density of S_i in $Q_j^{(i)}$.
3. Let $S \overset{\text{def}}{=} S_R$.

We will prove the following facts: (1) At the end of the execution, all positive sets are not singletons; (2) All the answers given to the algorithm throughout the execution are consistent with the final set S, up to an error of μ; and (3) The set S satisfies $|S| > 2^{n-1}$. Let us begin by a simple observation that will allow us to infer Fact (1):

Observation 11.1. *Let $Q \subseteq \{0,1\}^n$ be a positive set that was queried by the algorithm in iteration i (i.e., Q was answered by $\tilde{\nu}(Q) > 0$). Then $|Q| \geq \alpha^i \cdot 2^n$.*

Proof. In Step (2b) of iteration i we removed every queried set of size less than $\alpha^i \cdot 2^n$ from S, before answering the queries in Step (2c). Since Q is positive, it means that we answered the query Q with a non-zero value, and thus $|Q| \geq \alpha^i \cdot 2^n$. □

Claim 11.2. *Let Q be a positive set. Then $|Q| > 1$.*

Proof. By Observation 11.1, we have that $|Q| \geq \alpha^i \cdot 2^n$, for some $i \leq R$. However, by the hypothesis that $R < \frac{n}{\log((p+\mu)/\mu)} = \log_{1/\alpha}(2^n)$, we have that $\alpha^i \cdot 2^n \geq (1/\alpha)^{-R} \cdot 2^n > 1$. □

Let us now state another simple observation, which will be used to prove Facts (2) and (3).

Observation 11.3. *Let $Q \subseteq \{0,1\}^n$. Then, in any iteration $i \in [R]$, we remove less than $p \cdot \alpha^i \cdot 2^n$ strings from the intersection of Q with the tentative set; that is,*

$$|Q \cap S_{i-1}| - |Q \cap S_i| < p \cdot \alpha^i \cdot 2^n .$$

Proof. In Step (2b) of iteration i we remove strings from at most p sets from $T = S_{i-1}$ to obtain S_i, whereas each of these p sets is of size less than $\alpha^i \cdot 2^n$. □

The following claim asserts that for any query Q issued by A, the discrepancy between the answer that we give for Q and the density of the final set S in Q is at most μ. For the sake of streamlining the overall proof (of Theorem 11), we also define a fictitious "iteration zero": Recalling that $S_0 = \{0,1\}^n$, we consider a fictitious iteration $i = 0$, in which the algorithm queried $Q = \{0,1\}^n$, and received the answer $\tilde{\nu}(Q) = 1$. Then, the meaning of the following claim with respect to this query is that in the end of the execution, the density of the final set S in $\{0,1\}^n$ is at least $1 - \mu > 1/2$.

Claim 11.4. *For every iteration $0 \leq i \leq R$ and every query $Q = Q_j^{(i)}$, where $j \in [p]$, recall that $\tilde{\nu}(Q)$ denotes the answer given to the algorithm to query Q, and denote the density of the final set S in Q by $\nu(Q) = \frac{|Q \cap S|}{|Q|}$. Then, it holds that*

$$\left|\tilde{\nu}(Q) - \nu(Q)\right| = \frac{1}{|Q|} \cdot \left(\left||Q \cap S_i| - |Q \cap S|\right|\right) < \mu .$$

Proof. If $\tilde{\nu}(Q) = 0$, then $\nu(Q) = 0$ (since we only remove strings from the tentative set throughout the execution), and the claim follows. Also, if $i = R$, then the assertion in the claim is trivial (since $S_R = S$). Otherwise, it holds that

$$\begin{aligned}
|Q \cap S_i| - |Q \cap S| &= \sum_{j=i+1}^{R} |Q \cap S_{j-1}| - |Q \cap S_j| \\
&< \sum_{j=i+1}^{R} p \cdot \alpha^j \cdot 2^n && \text{(Obs. 11.3)} \\
&\leq \frac{|Q|}{\alpha^i} \cdot \sum_{j=i+1}^{R} p \cdot \alpha^j && \text{(Obs. 11.1)} \\
&= |Q| \cdot \sum_{j=1}^{R-i} p \cdot \alpha^j \\
&= |Q| \cdot \sum_{j=1}^{R-i} p \cdot \left(\frac{\mu}{p+\mu}\right)^j \\
&= |Q| \cdot p \cdot \frac{\mu}{p+\mu} \cdot \sum_{j=0}^{R-i-1} \left(\frac{\mu}{p+\mu}\right)^j \\
&< |Q| \cdot p \cdot \frac{\mu}{p+\mu} \cdot \frac{1}{1-\frac{\mu}{p+\mu}},
\end{aligned}$$

which equals $\mu \cdot |Q|$. It follows that $\left|\tilde{\nu}(Q) - \nu(Q)\right| < \mu$. □

Fact (2) follows immediately from Claim 11.4. To see that Fact (3) holds, invoke Claim 11.4 for $i = 0$, to get that $\left||S|/2^n - 1\right| = \left||S_0 \cap S|/|S| - 1\right| < \mu$, which implies that $|S| > (1-\mu) \cdot 2^n > 2^{n-1}$ (since $\mu < 1/2$). Since we reached a contradiction to the hypothesis that $R < \frac{n}{\log((p+\mu)/\mu)}$, we deduce that $R \geq \frac{n}{\log((p+\mu)/\mu)} > \frac{n}{\log(p+1)+\log(1/\mu)}$. ■

5.2 The lower bound holds even when S has density $1 - o(1)$

In this section we prove Theorem 3, which asserts a trade-off between the density of S and a lower bound on the number of iterations that hitters with oracle access to density estimators need 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 $\frac{\epsilon \cdot n}{\log(p(n)+1)+\log(1/\mu(n))}$.*

Proof. 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 $\alpha = \mu/(p+\mu)$. (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 $\mathcal{R} = \frac{n}{\log(p+1)+\log(1/\mu)}$ the number of iterations in the proof of Theorem 11, we will set the threshold ℓ_1 to the value that it obtained (in the proof of Theorem 11) 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 $\epsilon\cdot\mathcal{R} = \frac{\epsilon\cdot n}{\log(p+1)+\log(1/\mu)}$. Details follow.

Let $n \in \mathbb{N}$, and let $p = p(n)$ and $\mu = \mu(n)$. Let $\alpha = \mu/(p+\mu)$, and let $\mathcal{R} = \log_{1/\alpha}(2^n)$. Let A be a hitter as in the theorem's hypothesis, and assume towards a contradiction that A always uses $R < \epsilon\cdot\mathcal{R}$ iterations. We use the same adversarial simulation process as in the proof of Theorem 11, but with different parameters. Specifically, we start with $h_1 = 2^n$, but define $\ell_1 = \alpha^{R_1}\cdot 2^n$, where $R_1 = (1-\epsilon)\cdot\mathcal{R}+1$. Then, in each subsequent iteration $i \geq 2$, we define ℓ_i such that $\ell_i/\ell_{i-1} = \alpha$.

To see that in the end of the execution every positive set is not a singleton, observe that for every $i \in [R]$ it holds that $\ell_i = \ell_1\cdot\alpha^{i-1} = \alpha^{R_1+(i-1)}\cdot 2^n$. Thus, for every positive set Q that was queried in iteration $i \in [R]$ it holds that $|Q| \geq \ell_i \geq \ell_R = \alpha^{R_1+(R-1)}\cdot 2^n > \alpha^{\mathcal{R}}\cdot 2^n = 1$, where the last inequality is since $R < \epsilon\cdot\mathcal{R}$. The fact that our answer for every positive set is correct, up to an error of μ, can be proven almost identically to the proof in Theorem 11. [8] Finally, the overall number of strings that we removed from $S_0 = \{0,1\}^n$ throughout the execution in order to obtain the final set S is less than

$$\begin{aligned}\sum_{i=1}^{R} p\cdot\ell_i &= p\cdot\ell_1\cdot\sum_{i=1}^{R}\alpha^{i-1}\\ &< p\cdot\alpha^{R_1}\cdot 2^n\cdot\frac{1}{1-\alpha}\\ &< 2\cdot p\cdot\alpha^{(1-\epsilon)\cdot\mathcal{R}+1}\cdot 2^n\\ &= (2p\cdot\alpha)\cdot 2^{\epsilon\cdot n}\,,\end{aligned}$$

which is upper-bounded by $2^{\epsilon\cdot n}$ (since $p\cdot\alpha < \mu < 1/2$). ■

5.3 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

[8]For a positive set Q that was queried in iteration i we have that $|Q\cap S_i| - |Q\cap S| < \sum_{j=i+1}^{R} p\cdot\ell_j = \sum_{j=i+1}^{R} p\cdot\alpha^{R_1+j-1}\cdot 2^n \leq \frac{|Q|}{\alpha^{R_1+i-1}}\cdot\sum_{j=i+1}^{R} p\cdot\alpha^{R_1+j-1} = |Q|\cdot\sum_{j=1}^{R-i} p\cdot\alpha^j < \mu\cdot|Q|$.

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

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

In this section we prove lower bounds on the number of estimations needed to find a string in an unknown set S (with large density) when the estimation error μ is large (i.e., when $\mu \geq \frac{4 \cdot \log(n)}{n}$). We stress that in this setting we do not consider parallel algorithms, but rather focus only on the *total number of estimations* needed to solve the problem. Recall, from Section 4, that when $p = 1$ (i.e., when there is no parallelism), we refer to 1-parallel μ-error density estimators simply as μ-error density estimators.

The section is organized as follows. First, as a warm-up, we prove Item (1) of Theorem 7; that is, we show that finding a string in a set S with density less than $2 \cdot \mu$ requires $2^n/\text{poly}(n)$ queries. This proof will illustrate a basic idea that will appear in the subsequent proofs. Then, we will prove Theorem 6, which asserts that finding a string in a set S of size $|S| \geq 2^{n-1}$ requires $2^{\Omega(\mu \cdot n)}$ queries. Finally, we will prove Item (2) of Theorem 7, which generalizes Theorem 6, by showing a trade-off between the density of S and the number of estimations needed to find a string in S.

6.1 Warm-up: When the error is close to the density of S

Let us begin by proving a statement that implies both the "moreover" part of Theorem 6 and Item (1) of Theorem 7. Specifically, we show that for any value of μ (rather than $\mu \geq \frac{4 \cdot \log(n)}{n}$ or $\mu \geq \frac{12 \cdot \log(n)}{n}$, as in the statements of Theorems 6 and 7, respectively), finding a string in a set S with density $(2 - \Omega(1)) \cdot \mu$ requires $\Omega\left(\mu^2 \cdot 2^n/n\right)$ density estimations.

Proposition 14 *(a lower bound for small density). Let $\mu : \mathbb{N} \to (0, 1/2)$, and let A be a hitter with oracle access to μ-error density estimators. Then, for any constant $\beta < 2$, and sufficiently large $n \in \mathbb{N}$, there exists a set $S \subseteq \{0,1\}^n$ of size $|S| \geq \rho \cdot 2^n$, where $\rho = \beta \cdot \mu(n)$, 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 $\Omega\left(\mu(n)^2 \cdot 2^n/n\right)$.

Proof sketch. We present a high-level description of the proof, which, for simplicity, assumes that μ is constant. Since the low-level details of the full proof are relatively straightforward, we defer their presentation to Appendix B.

Similar to the proofs of Theorems 11 and 12, we simulate A for $R = o(2^n/n)$ iterations and provide adversarial oracle answers. However, in the current proof we use a threshold for defining "small" sets that is fixed throughout the execution, instead of iteratively decreasing the threshold (as in the previous proofs). Specifically, whenever A queries a set of size less than (roughly) n/μ^2, we provide the estimate zero; and whenever A queries a set of larger size, we provide the fixed estimate μ. Indeed, in the latter case, our estimate μ is equal to the estimation error. In the end of the execution, we let S be a random subset of density $\tau \in (\rho, 2\mu)$ of the set $\mathcal{L}$, where $\mathcal{L}$ consists of all elements that do not belong to any small queried set.

The idea behind this approach is as follows. First, observe that A did not find a positive singleton, since we answer zero for all small sets (and note that our answers are consistent with S, since the density of S in any small set is indeed zero). Now, in a large set Q, the density of S might be small, if many elements in Q also belong to small sets. However, since our estimate for Q is sufficiently close to zero (i.e., $\tilde{\nu}(Q)$ equals the estimation error μ), even if the actual density is zero (i.e., if $S \cap Q$ is empty), this estimate is still correct, up to an error of μ. On the other hand, the expected density of S in Q is at most $\tau < 2 \cdot \mu$, and thus, with very high probability, this density will not exceed $2 \cdot \mu$ (recall that $|Q| > n$). [9] To conclude the proof, note that the expected density of S in $\{0,1\}^n$ is $\tau \cdot |\mathcal{L}| > \rho \cdot 2^n$, where the inequality relies on the fact that $|\mathcal{L}| > 2^n - R \cdot n$ and on the choice of $\tau > \rho$. The full proof, which appears in Appendix B, merely details the above with more accurate parameters, in order to also handle a sub-constant μ. ■

6.2 The main lower bound

We now formally state Theorem 6 and prove it, by refining the basic idea that was used in the proof of Proposition 14. Note that we only have to prove the first part of the theorem's assertion (i.e., without the "moreover" part), since the "moreover" part follows from Proposition 14. We will actually prove a statement slightly stronger than the one in 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 15 *(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. 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*

[9] Specifically, we will use a Chernoff bound to claim that the density of S in Q deviates from τ by less than $2\mu - \tau < \mu \cdot (2-\beta)$, and hence (to have the probability of such a deviation be at most 2^{-n}) we will actually require that $|Q| > \frac{n}{(2-\beta)^2 \cdot \mu^2}$.

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. Let $\mu' = \mu/2$. We will first present an overview of the proof, where we assume for simplicity that μ is constant. Similar to the proof of Proposition 14, we simulate A, and answer its queries according to one fixed rule, which does not change throughout the execution. However, in the current proof, instead of using only one threshold for the size of "small" sets (as in the proof of Proposition 14), we consider several thresholds for the sizes of sets.

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 $\left[(i-1)\cdot\mu', i\cdot\mu'\right]$, 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; this is analogous to "small" sets in the proof of Proposition 14. 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 $(i-1) \cdot \mu'$. [10]

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 $(1-\mu) \cdot 2^n$. We now turn to the actual argument, which uses slightly more refined parameters, in order to also handle the case of a sub-constant μ.

The setting of parameters. Recall that $\mu' = \mu/2$, and, for simplicity, assume that $1/\mu'$ is an integer. We will repeatedly use the fact that

$$\mu' \geq \frac{2 \cdot \log(n)}{n} > 2^{-\log(n) + \log\log(n)} \geq 2^{-(\mu' \cdot n)/2 + \omega(1)} \, . \tag{6.1}$$

For $\alpha = 1/4\mu'$, we define the zero level to be $\mathcal{L}_0 = \{T \subseteq \{0,1\}^n : |T| \leq \alpha \cdot n\}$, and the first level to be $\mathcal{L}_1 = \{T \subseteq \{0,1\}^n : \alpha \cdot n < |T| \leq 2^{\mu' \cdot n}\}$. To see that $\mathcal{L}_1$ is non-empty, note that $\alpha \cdot n = n/4\mu' < 2^{\log(n) + (\mu' \cdot n)/2} \leq 2^{\mu' \cdot n}$, where the first inequality relies on Eq. (6.1), and the second inequality uses the fact that $\mu' \geq \frac{2 \cdot \log(n)}{n}$. Now, for every $i \in \{2, ..., 1/\mu'\}$, we define the i^{th} level to consist of all sets of size more than $2^{(i-1) \cdot \mu' \cdot n}$ and at most $2^{i \cdot \mu' \cdot n}$ (i.e., $\mathcal{L}_i = \{T \subseteq \{0,1\}^n : 2^{(i-1) \cdot \mu' \cdot n} < |T| \leq 2^{i \cdot \mu' \cdot n}\}$).

Assume towards a contradiction that the number of iterations used by A is at most $R = \frac{\mu'}{2} \cdot 2^{\mu' \cdot n} > 2^{\Omega(\mu \cdot n)}$, where the inequality relies on Eq. (6.1). The following fact follows from the definition of R and of the levels.

Fact 15.1. *For every $i \geq 3$ it holds that* $\min_{T \in \mathcal{L}_i}\{|T|\} > \left(\frac{2}{\mu'} \cdot R\right) \cdot \max_{T \in \mathcal{L}_{i-2}}\{|T|\}$.

Proof. Since $i - 2 \geq 1$, the maximal size of a set in $\mathcal{L}_{i-2}$ is $2^{(i-2) \cdot \mu' \cdot n}$. On the other hand, the minimal size of a set in $\mathcal{L}_i$ is more than $2^{(i-1) \cdot \mu' \cdot n} = \frac{2}{\mu'} \cdot R \cdot 2^{(i-2) \cdot \mu' \cdot n}$. □

The proof itself. We simulate A for R iterations, and for every query Q that it makes, we answer $\tilde{v}(Q) = i \cdot \mu'$, where i is such that $Q \in \mathcal{L}_i$. For the sake of streamlining the proof, we assume that A also queried the set $Q' = \{0,1\}^n \in \mathcal{L}_{1/\mu'}$, and received the estimate $\tilde{v}(Q') = 1$. Let $\tilde{S}$ be a random set such that every string $w \in \{0,1\}^n$ is included in $\tilde{S}$ with

[10] Note that the gap between our estimate of $i \cdot \mu'$ and the lower bound of $\delta_S(Q) \geq (i-2) \cdot \mu'$ is actually comprised of two gaps: The first is the gap between $i \cdot \mu'$ and $(i-1) \cdot \mu'$, which is because many strings $w \in Q$ might be included in S with probability $\ell(w) = i - 1$; and the second is the gap between $(i-1) \cdot \mu'$ and $(i-2) \cdot \mu'$, due to the deviation of the density of S from its expectation. This differs from the case of the upper bound of $\delta_S(Q) \leq (i+2) \cdot \mu'$, in which the gap of $2 \cdot \mu'$ is only due to the deviation of the density of S from its expectation.

probability $\ell(w) \cdot \mu'$, where $\ell(w)$ is the minimal $i \in \{0, ..., 1/\mu'\}$ such that w is included in a queried set $Q \in \mathcal{L}_i$.

Note that for any $Q \in \mathcal{L}_0$, and any choice of $\tilde{S}$, it holds that $\tilde{S} \cap Q = \emptyset$, and thus our estimate of $\tilde{\nu}(Q) = 0$ is correct. Now, fix a queried-set $Q \in \mathcal{L}_i$, where $i \geq 1$. The following claim asserts that with high probability, the density of $\tilde{S}$ in Q does not significantly exceed $i \cdot \mu'$.

Claim 15.2. *For $i \in [1/\mu']$, let $Q \in \mathcal{L}_i$ be a queried-set. Then, with probability more than $1 - 2^{-\mu' \cdot n}$, the density of $\tilde{S}$ in Q is at most $(i+2) \cdot \mu'$; that is, $\frac{|Q \cap \tilde{S}|}{|Q|} \leq (i+2) \cdot \mu'$.*

Proof. Every string $w \in Q$ is included in $\tilde{S}$ with probability $\ell(w) \cdot \mu' \leq i \cdot \mu'$ (where the inequality is since $\ell(w)$ is upper bounded by the level of Q, which is i). Thus, the expected density of $\tilde{S}$ in Q is at most $i \cdot \mu'$. Since $i \geq 1$, it holds that $|Q| > \alpha \cdot n = n/4\mu'$. Relying on a Chernoff bound, we have that $\Pr\left[\frac{|Q \cap \tilde{S}|}{|Q|} > (i+2) \cdot \mu'\right] < 2^{-(2\mu')^2 \cdot |Q|} < 2^{-\mu' \cdot n}$. □

Referring again to a fixed set $Q \in \mathcal{L}_i$, where $i \geq 1$, the following claim asserts that with high probability, the density of $\tilde{S}$ in Q is at least $(i-2) \cdot \mu'$.

Claim 15.3. *For $i \in [1/\mu']$, let $Q \in \mathcal{L}_i$ be a queried-set. Then, with probability more than $1 - 2^{-\mu' \cdot n}$, the density of $\tilde{S}$ in Q is at least $(i-2) \cdot \mu'$.*

Proof. When $i \leq 2$ the claim is trivial. For $i \geq 3$, the number of strings $w \in Q$ such that $\ell(w) \leq i-2$ is at most $R \cdot \max_{Q' \in \mathcal{L}_{i-2}}\{|Q'|\} < \frac{\mu'}{2} \cdot |Q|$, where the inequality is due to Fact 15.1. Now, note that every string w' such that $\ell(w') \geq i-1$ is included in $\tilde{S}$ with probability at least $(i-1) \cdot \mu'$. Thus, the expected density of $\tilde{S} \cap Q$ is at least

$$(1 - \mu'/2) \cdot (i-1) \cdot \mu' > (i - 1.5) \cdot \mu' ,$$

where the inequality is since $\mu' \cdot (i-1) \leq \mu' \cdot (1/\mu' - 1) < 1$.

Since $i \geq 3$, we have that $|Q| > 2^{2 \cdot \mu' \cdot n}$. Using a Chernoff bound, we get that $\Pr\left[\frac{|Q \cap \tilde{S}|}{|Q|} < (i-2) \cdot \mu'\right] < 2^{-(\mu'/2)^2 \cdot |Q|}$; to upper bound the right-hand side (i.e., the expression $2^{-(\mu'/2)^2 \cdot |Q|}$), note that

$$\begin{aligned} (\mu'/2)^2 \cdot |Q| &> 2^{-\mu' \cdot n + \omega(1) - 2} \cdot |Q| && \text{(Eq. (6.1))} \\ &> 2^{\mu' \cdot n} && (|Q| > 2^{2 \cdot \mu' \cdot n}) \\ &> \mu' \cdot n , \end{aligned}$$

which implies that $\Pr\left[\frac{|Q \cap \tilde{S}|}{|Q|} < (i-2) \cdot \mu'\right] < 2^{-\mu' \cdot n}$. □

Combining Claims 15.2 and 15.3, we deduce that for every queried-set $Q \in \mathcal{L}_i$, the probability that the density of $\tilde{S}$ in Q deviates from $i \cdot \mu'$ by more than $2 \cdot \mu' = \mu$ is less than $2 \cdot 2^{-\mu' \cdot n} < 1/R$. By a union-bound over R sets, there exists a choice of $\tilde{S}$ that satisfies the above for every queried-set, and we let S be such a choice. Since we assumed that A also issued the query $Q' = \{0,1\}^n \in \mathcal{L}_{1/\mu'}$ (and was answered by $\tilde{\nu}(Q') = 1$), it follows that the overall density of S in $\{0,1\}^n$ is at least $1 - \mu$. ■

6.3 A generalization: When S has density between 2μ and $1-\mu$

We now formally state and prove the lower bound in Item (2) of Theorem 7, which generalizes Theorem 15, by asserting a trade-off between the density of S and a lower bound on the number of estimations required to find a string in it. The proof will be obtained by slightly modifying the proof of Theorem 15. The corresponding upper bound in Item (2) of Theorem 7 is proved in Appendix A.2.

Theorem 16 *(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 $2^{\Omega((\mu/\rho)\cdot n)}$.*

Note that in Theorem 16 we actually consider densities in the interval $\rho \in [\mu, 1-\mu]$. Thus, there is a slight overlap between the lower bound in Theorem 16 and the stronger lower bound in Proposition 14, for density $\rho \leq (2-\Omega(1))\cdot\mu$.

Proof. In the proof of Theorem 15, 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 15, we will obtain a lower bound of about $2^{(\mu'/\rho)\cdot n}$ estimates, and the set S will be of density about $(\rho/\mu')\cdot\mu' = \rho$.

The parameter settings. Recall that $\mu' = \mu/2$, and assume for simplicity that ρ/μ' is an integer. Let $\eta = \rho/\mu' + 2$ be the number of non-zero levels that we will have. We will repeatedly use the following bounds that involve μ' and η:

Fact 16.1. *The following two inequalities hold:*

$$\mu' > 2^{-(\mu'\cdot n)/6}\,, \tag{6.2}$$

$$\frac{n}{\eta} > \frac{\mu'\cdot n}{2} \geq 3\cdot\log(n)\,. \tag{6.3}$$

Proof. By the hypothesis that $\mu \geq \frac{12\cdot\log(n)}{n}$, we have that $\mu' \geq \frac{6\cdot\log(n)}{n} > 2^{-\log(n)} \geq$

$2^{-(\mu' \cdot n)/6}$. To lower bound $\frac{n}{\eta}$, note that

$$\frac{n}{\eta} = \frac{n}{\rho/\mu' + 2} \geq \frac{n}{2 \cdot (\rho/\mu')} \geq \frac{\mu' \cdot n}{2} \geq 3 \cdot \log(n) ,$$

where the first inequality is because $\rho/\mu' \geq 2$ (since $\rho \geq \mu = 2 \cdot \mu'$), and the last inequality uses the fact that $\mu' \geq 6 \cdot \log(n)/n$. $\square$

For $\alpha = 1/(4 \cdot \mu'^2 \cdot \eta)$, we define the zero level to be $\mathcal{L}_0 = \{T \subseteq \{0,1\}^n : |T| \leq \alpha \cdot n\}$, and the first level to be $\mathcal{L}_1 = \{T \subseteq \{0,1\}^n : \alpha \cdot n < |T| \leq 2^{n/\eta}\}$. Note that the first level is non-empty, because $\alpha \cdot n = \frac{n}{4 \cdot \mu'^2 \cdot \eta}$, and

$$\begin{aligned} \frac{n}{4 \cdot \mu'^2 \cdot \eta} &< \frac{n}{\mu'^2} && (\eta > 1) \\ &< 2^{\mu' \cdot n/3 + \log(n)} && (\text{Eq. } (6.2)) \\ &< 2^{n/\eta} . && (\text{Eq. } (6.3)) \end{aligned}$$

For every $i \in \{2, ..., \eta\}$, we define the i^{th} level to consist of all sets of size more than $2^{(i-1) \cdot n/\eta}$ and at most $2^{i \cdot n/\eta}$. Assume towards a contradiction that the number of iterations used by A is at most $R = \frac{\mu'}{2} \cdot 2^{n/\eta} > 2^{n/\eta - (\mu' \cdot n)/6 - 1} \geq 2^{\Omega((\mu/\rho) \cdot n)}$, where the first inequality relies on Eq. (6.2), and the second inequality relies on Eq. (6.3). Similarly to Fact 15.1, 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 $\frac{2}{\mu'} \cdot R$.

The proof itself. We simulate A for R iterations, and for every query Q that it makes, we answer $\tilde{v}(Q) = i \cdot \mu'$, where i is such that $Q \in \mathcal{L}_i$. In the end of the execution, let $\tilde{S}$ be a random set such that every string $w \in \{0,1\}^n$ is included in $\tilde{S}$ with probability $\ell(w) \cdot \mu'$ (where $\ell(w)$ is defined as in the proof of Theorem 15). [11] Let us now fix a queried-set Q, and prove that $\tilde{S}$ is consistent with our estimate $\tilde{v}(Q)$, with high probability.

Claim 16.2. *For $i \in [1/\mu']$, let $Q \in \mathcal{L}_i$ be a queried-set. Then, with probability more than $1 - 2^{-n/\eta+1}$, the density of $\tilde{S}$ in Q is at least $(i-2) \cdot \mu'$ and at most $(i+2) \cdot \mu'$.*

Proof. The upper bound is proved similarly to the proof of Claim 15.2, relying on the fact that $|Q| > \alpha \cdot n$ (since $i \geq 1$), and using a Chernoff bound to deduce that $\Pr\left[|Q \cap \tilde{S}|/|Q| > (i+2) \cdot \mu'\right] < 2^{-(2\mu')^2 \cdot |Q|} < 2^{-(2\mu')^2 \cdot \alpha \cdot n} = 2^{-n/\eta}$.

As for the lower bound, when $i \leq 2$ it is trivial. When $i \geq 3$, using an argument analogous to the one in the proof of Claim 15.3, we infer that the number of strings $w \in Q$ such that $\ell(w) \leq i-2$ is at most $R \cdot \max_{Q' \in \mathcal{L}_{i-2}} |Q'| < \frac{\mu'}{2} \cdot |Q|$, and thus the expected density of $\tilde{S} \cap Q$ is at least $(i - 1.5) \cdot \mu'$. Since $i \geq 3$, we have that $|Q| > 2^{2 \cdot n/\eta}$, and using

[11]Here we use the fact that $\rho \leq 1 - \mu$, to avoid outputting estimations that are larger than 1, or taking probabilities that are larger than 1.

a Chernoff bound, we get that $\Pr\left[\frac{|Q\cap\tilde{S}|}{|Q|} < (i-2)\cdot\mu'\right] < 2^{-(\mu'/2)^2\cdot|Q|} < 2^{-2^{2\cdot n/\eta-2\cdot\log(4/\mu')}}$. To bound this expression, note that

$$\begin{aligned}
2^{2\cdot n/\eta-2\cdot\log(4/\mu')} &> 2^{2\cdot(n/\eta)-(\mu'\cdot n)/6-2} && \text{(Eq. (6.2))}\\
&> 2^{2\cdot(n/\eta)-(\mu'\cdot n)/2} && (\mu'\cdot n=\omega(1))\\
&> 2^{n/\eta} && \text{(Eq. (6.3))}\\
&> n/\eta\,,
\end{aligned}$$

which implies that $\Pr\left[\frac{|Q\cap\tilde{S}|}{|Q|} < (i-2)\cdot\mu'\right] < 2^{-n/\eta}$. □

Relying on Claim 16.2, and applying a union-bound over R sets, there exists a choice of $\tilde{S}$ that satisfies the above for every queried-set, and we let S be such a choice. The overall density of S in $\{0,1\}^n$ is at least $(\eta-2)\cdot\mu'=\rho$. ■

7 A lower bound on the number of density queries when there is no estimation error

In this section we show that hitters with oracle access to zero-error density estimators need $n-O(1)$ density queries in order to find a string in a set S with constant density. We actually show a tight lower bound on the number of queries that such algorithms need, where this lower bounds depends on the exact density of S.

Theorem 17 *(a lower bound on the number of queries when there is no estimation error; Theorem 8, restated). Let A be a hitter with oracle access to zero-error density estimators (i.e., $\mu=0$). Then, for any $n\in\mathbb{N}$, and any density $\rho\in(0,1)$, there exists a set $S\subseteq\{0,1\}^n$ of size $|S|>\rho\cdot 2^n$, and a zero-error density estimator f_S for S, such that the number of queries that A uses when given oracle access to f_S is at least $n-\lfloor\log(1/(1-\rho))\rfloor-1$.*

Note that for any density $\rho<1/2$, the lower bound in Theorem 17 is simply $n-1$ queries. As mentioned in Section 1.3, the lower bound in Theorem 17 is tight up to a single bit, due to the following algorithm. Given a guarantee that $|S|>\rho\cdot 2^n$, for some $\rho\in(0,1)$, the algorithm can disregard arbitrary $\rho\cdot 2^n$ elements of the search space (e.g., the last ones lexicographically), and use the method of conditional probabilities to find a string $s\in S$ among the remaining $(1-\rho)\cdot 2^n$ elements. The number of queries that such algorithm uses is $\lceil\log((1-\rho)\cdot 2^n)\rceil = n-\lfloor\log(1/(1-\rho))\rfloor$.

Proof. Similar to previous proofs, we will simulate A and provide adversarial answers. Throughout the execution, we will maintain a *template* for S: This is a partition of $\{0,1\}^n$, where each set P in the partition is labeled with the (exact) density of S in P. The partition starts out only with the set $\{0,1\}^n$, which is labeled with the overall density S is guaranteed to have. Given each query Q of A, we will refine the existing partition, by splitting each set P in the partition to $P\cap Q$ and $P\setminus Q$, and labelling each of these two parts with a corresponding density of S. Then, we will answer A's query (i.e., output the density of S in Q) according to the updated template. Indeed, we have not yet specified exactly *how* we split the density of S in P between $P \cap Q$ and $P \setminus Q$, for each P; for the moment, just assume that this is done in a way consistent with the density of S in P.

Since we only refine the partition throughout the execution, this approach yields answers to A that are perfectly consistent with any set S that is constructed according to the template. [12] Our main challenge is to prove that this approach prevents A from finding a positive singleton using few queries. To this end, note that if A queried a singleton and received a positive answer, then one of the sets in our partition is a singleton, labeled with density 1. Thus, it suffices for us to prove that after less than $n - \log(1/(1-\rho))$ queries, the partition does not contain a set that is labeled with density 1 (i.e., a set that is fully contained in S).

This will be done by showing an appropriate method to split, for any given set P in a partition and query Q, the $|P \cap S|$ strings of S in P, between $P \cap Q$ and $P \setminus Q$. Specifically, our method will maintain the following invariant: Denote by m the number of strings in P that are *not* in S, and assume for simplicity that m is an even number. We will ensure that if $P \cap Q$ contains any string from S, then it contains at least $m/2$ strings that are *not* in S, and ditto for $P \setminus Q$. Since the initial set $\{0,1\}^n$ has $(1-\rho) \cdot 2^n = 2^{n-1/(1-\rho)}$ strings that are not in S, after the i^{th} query we have that *every set in the partition that contains a string from S, also contains at least $2^{n-1/(1-\rho)-i}$ strings that are not in S.* Hence, the algorithm will need more than $n - \log(1/(1-\rho))$ queries to obtain a set that is fully contained in S.

The specific method that yields the invariant above is as follows: If both $P \cap Q$ and $P \setminus Q$ are of size at least $m/2$, then we allocate the m strings that are *not* in S such that both sets contain $m/2$ such strings. Otherwise, we allocate the strings such that the smaller of the two sets (which is of size less than $m/2$) contains only strings that are not in S; and it follows that the larger set contains more than $m/2$ strings that are not in S. We now provide the full details, which are slightly more cumbersome, since they also account for the case when m is odd.

The proof details. We formalize the approach above by defining a potential function Φ over the sets in a partition of $\{0,1\}^n$ (in which each set is labeled by the density of S in it) as follows: The potential of a set P is $\Phi(P) = |P| - |P \cap S| + 1$, if $|P \cap S| > 0$; and the potential is $\Phi(P) = \infty$, if $|P \cap S| = 0$ (where the latter case also includes the case in which P is empty). That is, the potential of a non-empty set P is the *number of strings in P that are not in S,* plus 1; or ∞, if no string in P is in S. Note that if a set P in the partition is a positive singleton, then $\Phi(P) = 1$. Thus, to show that after i iterations the algorithm did not find a positive singleton, it suffices to show that in the end of the i^{th} iteration, for every set P in the corresponding partition it holds that $\Phi(P) > 1$.

Recall that we want to construct a set S of density larger than ρ. In the beginning of the algorithm's execution, the partition is comprised only of the set $\{0,1\}^n$, and we define the density of S in $\{0,1\}^n$ to be exactly $\rho' = \rho + 2^{-n}$. Thus, $\Phi(\{0,1\}^n) =$

[12]For example, in the end of the execution, we can construct S by including in it, for each set P in the final partition, the first lexicographically $|P \cap S|$ strings from P.

$2^n - \rho' \cdot 2^n + 1 = 2^{n-\log(1/(1-\rho))}$. Given each query Q of A, and for each set P in the partition, we will rely on the following claim:

Claim 17.1. *Let $P \subseteq \{0,1\}^n$ such that the density of S in P is determined (i.e., P is labeled with the value $|P \cap S|$), and let $Q \subseteq \{0,1\}^n$. Then, there exists a way to label $P \cap Q$ and $P \setminus Q$ with densities of S that are consistent with the density of S in P, such that $\Phi(P \cap Q)$ and $\Phi(P \setminus Q)$ are lower bounded by $\Phi(P)/2$.*

Proof. Let $T_1 = P \cap Q$ and $T_2 = P \setminus Q$. It will be useful for us to think about allocating the strings that are *not* in S to T_1 and to T_2 (instead of allocating the strings that are in S to the two subsets). To this end, denote by $m = \Phi(P) - 1$ the number of strings in P that are not in S. Our goal is to allocate these m strings such that, for $i = 1, 2$, it holds that $\Phi(T_i)$ will be lower bounded by $\Phi(P)/2$. This will happen if either all the strings in T_i are not in S, in which case $\Phi(T_i) = \infty$; or the number of strings in T_i that are not in S is at least $\Phi(P)/2 - 1 = \frac{m-1}{2}$ (which implies that $\Phi(T_i) \geq \Phi(P)/2$).

Our allocation rule is as follows. If one set (say, T_1) is of size less than $\lfloor m/2 \rfloor$, then the other set (say, T_2) is necessarily of size more than $\lceil m/2 \rceil$ (because $|P| \geq m$). In this case, we let $T_1 \cap S = \emptyset$; that is, all the strings in T_1 are not in S, which implies that $\Phi(T_1) = \infty$. It follows that the number of strings in T_2 that are not in S is $m - |T_1| > m/2 > \frac{m-1}{2}$. Otherwise, both T_1 and T_2 are of size at least $\lfloor m/2 \rfloor$. In this case, we allocate $\lfloor m/2 \rfloor$ strings that are not in S to T_1, and $\lceil m/2 \rceil$ strings that are not in S to T_2, and rely on the fact that $\lceil m/2 \rceil \geq \lfloor m/2 \rfloor \geq \frac{m-1}{2}$. □

Given each query Q and set P, we allocate the $|P \cap S|$ strings to $P \cap Q$ and to $P \setminus Q$ as in Claim 17.1. It follows that after the i^{th} iteration, the potential of each set in the partition is at least $2^{n-\log(1/(1-\rho))-i}$. Hence, at least $n - \lfloor \log(1/(1-\rho)) \rfloor$ queries are required in order to find a positive singleton. Omitting the requirement that the algorithm actually queries the string $s \in S$ that it found (recall that this requirement is convenient for the proof, but was not required in Definition 10), we deduce that any algorithm needs at least $n - \lfloor \log(1/(1-\rho)) \rfloor - 1$ queries to solve the problem. ■

8 An open question: A strong version of the [MNN94] conjecture

The results in Section 1.1 affirm a weak version of the conjecture of Motwani, Naor, and Naor [MNN94] (see Section 2), but fall short of proving the strongest possible version of this conjecture, which asserts the following.

Conjecture 1 *(a strong version of the conjecture of [MNN94]). Consider an unknown set of solutions $S \subseteq \{0,1\}^n$ of size $|S| \geq 2^{n-1}$, and algorithms that, in each iteration, can obtain the* exact density *of S in p subsets of $\{0,1\}^n$ of their choice, in parallel. The conjecture is that such algorithms require $\Omega(n/\log(p+1))$ iterations to find a string in $s \in S$.*

Note that Theorem 8 proves the special case of Conjecture 1 for $p = 1$ (i.e., when there is no parallelism). The theorem also implies a lower bound of $\frac{n-2}{p}$ iterations for parallel algorithms with no estimation error (otherwise, a non-parallel algorithm could simulate the parallel algorithm and solve the problem using less than $n-2$ queries). We mention that even proving Conjecture 1 for restricted types of algorithms (e.g., computationally bounded algorithms, or algorithms that are restricted to "simple" types of queries) would be interesting.

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

References

[Gol11] Oded Goldreich. In a world of P=BPP. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay Randomness and Computation*, pages 191–232. 2011.

[GW14] Oded Goldreich and Avi Widgerson. On derandomizing algorithms that err extremely rarely. In *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 109–118. 2014.

[IKW02] 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.

[IW98] 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.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KUW88] Richard M. Karp, Eli Upfal, and Avi Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36(2):225–253, 1988.

[MNN94] 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.

[Tel17a] 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.

[Tel17b] Roei Tell. Quantified derandomization of linear threshold circuits. *Electronic Colloquium on Computational Complexity: ECCC*, 24:145, 2017.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[Vad12] Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

[Vio05] Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal of Computing*, 42(3):1218–1244, 2013.

Appendix A Two upper bounds

In the current appendix we prove two upper bounds: Theorem 1, for the setting of parallel algorithms, and Theorem 5, for the setting of a large estimation error. Both algorithms follow a similar approach: They operate in *iterations*, where in each iteration they equipartition the current search space, obtain estimates for all sets in the partition, and recurse into the set with the highest estimated density. The difference between the algorithms lies in the parameters and in some low-level details.

A.1 An upper bound for parallel algorithms.

We state Theorem 1 formally, using the definitions from Section 4.

Theorem 18 *(an upper bound for parallel algorithms; Theorem 1, restated). Let $p : \mathbb{N} \to \mathbb{N}$ and $\mu : \mathbb{N} \to \mathbb{R}^+$ such that $\mu(n) < \frac{\lfloor \log(p(n)+1) \rfloor}{4n}$. Then, there exists a hitter with oracle access to a p-parallel μ-error density estimator, denoted A, such that for any $S \subseteq \{0,1\}^n$ satisfying $|S| \geq 2^{n-1}$, and any $p(n)$-parallel $\mu(n)$-error density estimator f_S for S, when A is given access to f_S, then A finds a string $s \in S$ after* at most $\left\lceil \frac{n}{\lfloor \log(p(n)+1) \rfloor} \right\rceil$ iterations.

Proof. Let $n \in \mathbb{N}$, and let $p = p(n)$ and $\mu = \mu(n)$. For any string $w \in \{0,1\}^k$, where $k \leq n$, denote by $C_w \subseteq \{0,1\}^n$ the subcube that consists of all strings that start with the prefix w; that is, $C_w = \{w \circ x : x \in \{0,1\}^{n-|w|}\}$. Let $R = \lceil n / \lfloor \log(p+1) \rfloor \rceil$.

The algorithm constructs a string $s \in S$ in R iterations, where in each iteration a prefix of s is extended by $\lfloor \log(p+1) \rfloor$ bits. Specifically, the algorithm initializes s_0 as the empty string, and for every iteration $i = 1, ..., R$, acts as follows:

1. Let $\beta = \min\{\lfloor \log(p+1) \rfloor, n - |s_{i-1}|\}$. Send the following $2^\beta - 1 \leq p$ queries to the oracle: For every $w \in \{0,1\}^\beta$ such that $w \neq 1^\beta$, the query-tuple includes the query $C_{s_{i-1} \circ w}$ (i.e., the queries are $\left\{C_{s_{i-1} \circ w} : w \in \{0,1\}^\beta \setminus \{1^\beta\}\right\}$).
2. If there exists $w \in \{0,1\}^\beta \setminus \{1^\beta\}$ such that $\tilde{\nu}(C_{s_{i-1} \circ w}) \geq \frac{1}{2} - 2 \cdot \mu \cdot (i-1) - \mu$, then let $s_i = s_{i-1} \circ w$ (if there exist several such w's, pick one arbitrarily). Otherwise, let $s_i = s_{i-1} \circ 1^\beta$.

In the end of the execution, the algorithm outputs $s = s_R$.

Since for every $i \in [R-1]$ it holds that $|s_i| = |s_{i-1}| + \lfloor \log(p+1) \rfloor$, after R iterations we have that $|s| = n$. The main claim in the analysis is the following:

Claim 18.1. *For every $i \in \{0, ..., R-1\}$ it holds that the density of S in C_{s_i} is at least $\frac{1}{2} - 2 \cdot \mu \cdot i$.*

Proof. The proof is by induction on i. The base, when $i = 0$, follows by the hypothesis that $|S| \geq 2^{n-1}$. For the induction step, we assume that the claim holds for a generic $i < R-1$, and show that it also holds for $i+1$. In iteration $i+1$, if at least one of the p answers returned by the oracle is at least $1/2 - 2 \cdot \mu \cdot i - \mu$, then in Step 2(b) the algorithm will define $s_i = s_{i-1} \circ w$, where $w \neq 1^{\lfloor \log(p+1) \rfloor}$ is such that $\tilde{\nu}(C_{s_i \circ w}) \geq 1/2 - 2 \cdot \mu \cdot i - \mu$. Since the estimation error is at most μ, it follows that in this case, the density of S in $C_{s_{i+1}} = C_{s_i \circ w}$ is at least $1/2 - 2 \cdot \mu \cdot (i+1)$.

Otherwise, if the answers to all the p queries are less than $1/2 - 2 \cdot \mu \cdot i - \mu$, then the density of S in each of the corresponding p subcubes is less than $1/2 - 2 \cdot \mu \cdot i$. However, by the induction hypothesis, the density of S in C_{s_i} is at least $1/2 - 2 \cdot \mu \cdot i$. Since the collection $\{C_{s_i \circ w} : w \in \{0,1\}^{\lfloor \log(p+1) \rfloor}\}$ is a partition of C_{s_i}, it follows that the density of S in $C_{s_{i+1}} = C_{s_i \circ 1^{\lfloor \log(p+1) \rfloor}}$ is more than $1/2 - 2 \cdot \mu \cdot i > 1/2 - 2 \cdot \mu \cdot (i+1)$. □

Claim 18.1 implies that the density of S in $C_{s_{R-1}}$ is at least $\frac{1}{2} - 2 \cdot \mu \cdot (R-1) > \frac{1}{2} - \frac{2 \cdot \mu \cdot n}{\lfloor \log(p+1) \rfloor} > 0$, where the last inequality is due to our hypothesis that $\mu < \frac{\lfloor \log(p+1) \rfloor}{4n}$. It follows that $C_{s_{R-1}} \cap S \neq \emptyset$. Now, in the R^{th} iteration, the algorithm queries all the singletons in $C_{s_{R-1}}$, and thus it will indeed find $s_R \in S$. ■

A.2 An upper bound for algorithms with a large estimation error

We now prove a statement that implies both Theorem 5 and the upper bound in Item (2) of Theorem 7.

Theorem 19 *(an upper bound for algorithms with a large estimation error; Theorem 5, restated). Let $\mu : \mathbb{N} \to (0, 1/2)$, and let $\rho : \mathbb{N} \to \mathbb{R}^+$. Then, there exists a hitter with oracle access to a μ-error density estimator, denoted A, such that for any $S \subseteq \{0,1\}^n$ of size $|S| \geq \rho(n) \cdot 2^n$, and any $\mu(n)$-error density estimator f_S for S, when A is given access to f_S, then A finds a string $s \in S$ using less than $\frac{2}{\mu(n)} \cdot 2^{(2\mu(n)/\rho(n)) \cdot n}$* estimates.

As mentioned in Theorem 5, when $\mu \geq \log(n)/n$, the upper bound in Theorem 19 is less than $2^{O((\mu/\rho) \cdot n)}$. This is the case because $\mu \geq \log(n)/n$ implies that $1/\mu < 2^{\log(n)} \leq 2^{\mu \cdot n} < 2^{(\mu/\rho) \cdot n}$, and thus $\frac{2}{\mu} \cdot 2^{(2\mu/\rho) \cdot n} < 2^{(3\mu/\rho) \cdot n + 1} = 2^{O((\mu/\rho) \cdot n)}$.

Proof. Let $n \in \mathbb{N}$, and let $\mu = \mu(n)$ and $\rho = \rho(n)$. When $\rho \leq 2\mu$, the upper bound is trivial (since it is larger than 2^n), and thus we assume that $\rho > 2\mu$. Similarly to the proof of Theorem 18, the algorithm that yields the upper bound constructs $s \in S$ in iterations, where in each iteration a prefix for s is extended by $\lceil (2\mu/\rho) \cdot n \rceil$ bits. Thus, the total number of iterations is $R = n / \lceil (2\mu/\rho) \cdot n \rceil < \rho/2\mu$. Now, as in the proof of Theorem 18, in each iteration, after extending the prefix for s, the density of S in the resulting subcube might be lower by 2μ than the density of S in the subcube corresponding to the non-extended prefix. However, since the number of iterations is less than $\rho/2\mu$, the total "loss" of density is less than ρ, which implies that the final string is in S. Details follow.

Let $\beta = \lceil (2\mu/\rho) \cdot n \rceil$ be the number of bits by which the prefix of s is extended in each iteration, and let s_0 be the empty string. Let $R = \lceil n/\beta \rceil$. For $i = 1, ..., R$, the algorithm acts as follow:

1. For every $w \in \{0,1\}^{\min\{\beta, n-|s_{i-1}|\}}$, send the query $C_{s_{i-1} \circ w}$ to the oracle.
2. Let w' be such that the estimate $\tilde{\nu}(C_{s_{i-1} \circ w'})$ is maximal among the received answers. Then, let $s_i = s_{i-1} \circ w'$.

In the end of the execution, the algorithm outputs $s = s_R$.

Since for every $i \in [R-1]$ it holds that $|s_i| = |s_{i-1}| + \beta$, after R iterations we have that $|s_R| = n$. Also note that for every $i \in \{0, ..., R-1\}$, the density of S in C_{s_i} is at least $\rho - 2 \cdot \mu \cdot i$; the proof of this fact is analogous to the proof of Claim 18.1. [13] Thus, the density of S in $C_{s_{R-1}}$ is at least $\rho - 2 \cdot \mu \cdot (R-1) > \rho - 2 \cdot \mu \cdot (\rho/2\mu) = 0$, which implies that $C_{s_{R-1}} \cap S \neq \emptyset$. Now, in the R^{th} iteration, the algorithm queries all the singletons in $C_{s_{R-1}}$, and thus (since $\mu < 1/2$) it will indeed find $s_R \in S$. The overall number of density estimations used by the algorithm is

$$R \cdot 2^\beta < \left(\frac{\rho}{2\mu} + 1 \right) \cdot 2^{(2\mu/\rho) \cdot n + 1} < \frac{2}{\mu} \cdot 2^{(2\mu/\rho) \cdot n} . \qquad \blacksquare$$

A.3 On the "loss" of 2μ in density in each iteration

In the two algorithms described in the proofs of Theorems 18 and 19, in each iteration, the guarantee on the density of S in the "current" search space (i.e., in the subcube C_{s_i}) decreases by a value of 2μ. We note that this loss is essentially unavoidable, in the worst case, when only obtaining estimations of the density of S in the sets of an equipartition. To see this, consider the following setting.

Fix an arbitrary equipartition of $\{0,1\}^n$ to $p = \omega(1)$ sets. Let $\epsilon = 2\mu/p = o(\mu)$, and define S such that the density of S in each of the first $(p-1)$ sets is $1/2 + \epsilon$, and

[13] Specifically, we use induction on i. The base case $i = 0$ holds due to the hypothesis that $|S| \geq \rho \cdot 2^n$. For the induction step (assuming that the claim holds for $i < R$, and proving for $i+1$), denote $\eta = \rho - (i-1) \cdot 2\mu$. Due to the induction hypothesis, the density of S in at least one subcube $C_{s_{i_1} \circ w}$ will be η or more, which implies that the estimate $\tilde{\nu}(C_{s_{i-1} \circ w})$ will be at least $\eta - \mu$. Hence, the algorithm will let $s_i = s_{i-1} \circ w'$, where w' is such that $\tilde{\nu}(C_{s_{i-1} \circ w'}) \geq \eta - \mu$. It follows (since the estimation error is μ) that the density of S in $C_{s_{i-1} \circ w'}$ is at least $\eta - 2\mu$.

the density of S in the p^{th} set is $1/2 - 2\mu + \epsilon$. Note that the density of S in $\{0,1\}^n$ is $\frac{p-1}{p} \cdot (1/2 + \epsilon) + \frac{1}{p} \cdot (1/2 - 2\mu + \epsilon) = 1/2$. However, in this setting, even if the algorithm queries all p sets, an oracle might return an estimation of $1/2 - \mu + \epsilon$ for each of the sets. Thus, the algorithm might recurse into the p^{th} set, with density $1/2 - 2\mu + \epsilon = 1/2 - (2 - o(1)) \cdot \mu$.

Appendix B Proof details for Proposition 14

Let us restate Proposition 14 and prove it in full.

Proposition 20 *(a lower bound for small density; Proposition 14, restated). Let $\mu : \mathbb{N} \to (0, 1/2)$, and let A be a hitter with oracle access to μ-error density estimators. Then, for any constant $\beta < 2$, and sufficiently large $n \in \mathbb{N}$, there exists a set $S \subseteq \{0,1\}^n$ of size $|S| \geq \rho \cdot 2^n$, where $\rho = \beta \cdot \mu(n)$, 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 $\Omega\left(\mu(n)^2 \cdot 2^n/n\right)$.*

Proof. For $\mu \leq \sqrt{n/2^n}$, the asserted lower bound is trivial; we thus assume that $\mu > \sqrt{n/2^n}$. Let A be a hitter as in the theorem's hypothesis, and let $\alpha = 16/\left(2 \cdot \mu - \rho\right)^2$. Assume towards a contradiction that A uses at most $R = \frac{1}{\alpha} \cdot \frac{2 \cdot \mu - \rho}{2 \cdot \mu + \rho} \cdot (2^n/n) = \Omega(\mu^2 \cdot (2^n/n))$ iterations. We simulate A, and for each query Q that it sends, if $|Q| \geq \alpha \cdot n$, then our answer is $\tilde{\nu}(Q) = \mu$, and otherwise (if $|Q| < \alpha \cdot n$), our answer is $\tilde{\nu}(Q) = 0$. Let $\mathcal{L}_0$ be the union of all queried-sets of size smaller than $\alpha \cdot n$, and let $\mathcal{L} = \{0,1\}^n \setminus \mathcal{L}_0$. Let $\tilde{S}$ be a random subset of $\mathcal{L}$ such that every element in $\mathcal{L}$ is included in $\tilde{S}$ independently with probability $\tau = \rho + \frac{3}{4} \cdot (2 \cdot \mu - \rho)$.

Note that for any choice of $\tilde{S}$, and any queried-set Q such that $|Q| < \alpha \cdot n$, our answer $\tilde{\nu}(Q) = 0$ is consistent with $\tilde{S}$ (because $\tilde{S} \cap \mathcal{L}_0 = \emptyset$). The following claim implies that with high probability, all of our answers for larger sets are also consistent with $\tilde{S}$, up to an error of μ.

Claim 20.1. *With probability at least $2/3$, for every queried-set Q such that $|Q| \geq \alpha \cdot n$ we have that $\frac{|Q \cap \tilde{S}|}{|Q|} \leq 2 \cdot \mu$.*

Proof. Let Q be a queried-set such that $|Q| \geq \alpha \cdot n$. Note that every string $w \in Q$ is included in $\tilde{S}$ with probability either τ or 0 (the latter case happens if w belongs to a small queried-set). Thus, the expected density of $\tilde{S}$ in Q is at most $\tau < 2 \cdot \mu$. Relying on a Chernoff bound, we have that $\Pr\left[\frac{|Q \cap \tilde{S}|}{|Q|} \geq 2 \cdot \mu\right] < 2^{-(2 \cdot \mu - \tau)^2 \cdot |Q|} \leq 2^{-\frac{1}{16}(2 \cdot \mu - \rho)^2 \cdot \alpha \cdot n}$, which equals 2^{-n}, since $\alpha = 16/\left(2 \cdot \mu - \rho\right)^2$. The claim follows by a union bound over the number of queries-sets of size at least $\alpha \cdot n$, which is at most $R = o(2^n)$. $\square$

The following claim asserts that with high probability, $\tilde{S}$ has density at least ρ.

Claim 20.2. *With probability at least $2/3$, the set $\tilde{S}$ satisfies $|\tilde{S}| \geq \rho \cdot 2^n$.*

Proof. First note that $|\mathcal{L}| > \frac{2\cdot\rho}{2\cdot\mu+\rho}\cdot 2^n$. To see that this holds, observe that $\mathcal{L} = \{0,1\}^n \setminus \mathcal{L}_0$, where $\mathcal{L}_0$ is the union of at most R sets, each of them of size less than $\alpha \cdot n$. Thus, $|\mathcal{L}|$ is lower bounded by $2^n - R\cdot(\alpha\cdot n) = \left(1 - \frac{2\cdot\mu-\rho}{2\cdot\mu+\rho}\right)\cdot 2^n = \frac{2\cdot\rho}{2\cdot\mu+\rho}\cdot 2^n$.

Now, relying on a Chernoff bound, for any $\eta > 0$, the probability that the size of $\tilde{S}$ is less than $(\tau-\eta)\cdot|\mathcal{L}|$ is at most $2^{-\eta^2\cdot|\mathcal{L}|}$. In particular, for $\eta = \frac{2\cdot\mu-\rho}{4}$, we have that $(\tau-\eta)\cdot|\mathcal{L}|$ equals

$$\begin{aligned}\left(\tau - \frac{2\cdot\mu-\rho}{4}\right)\cdot|\mathcal{L}| &= \frac{1}{2}\cdot(2\cdot\mu+\rho)\cdot|\mathcal{L}| \\ &> \frac{1}{2}\cdot(2\cdot\mu+\rho)\cdot\left(\frac{2\cdot\rho}{2\cdot\mu+\rho}\cdot 2^n\right) \\ &= \rho\cdot 2^n\ , \end{aligned} \tag{B.1}$$

and the probability of $\tilde{S}$ being of size less than the expression in Eq. (B.1) is at most $2^{-\frac{(2\cdot\mu-\rho)^2}{16}\cdot|\mathcal{L}|} < 2^{-\Omega(\mu^2\cdot 2^n)} < 2^{-\Omega(n)}$, where the last inequality relied on the fact that $\mu > \sqrt{n/2^n}$. □

Overall, with positive probability over choice of $\tilde{S}$ it holds that our answers to all the queries of A are consistent with $\tilde{S}$, up to an error of μ, and that $|\tilde{S}| \geq \rho\cdot 2^n$. We take S to be any set that satisfies these two conditions. ■