Finding Heavy Hitters from Lossy or Noisy Data

Lucia Batman, Russell Impagliazzo*, Cody Murray**,
and Ramamohan Paturi**

Department of Computer Science and Engineering,
University of California, San Diego

Abstract. Motivated by Dvir et al. and Wigderson and Yehudayoff [3,10], we examine the question of discovering the set of heavy hitters of a distribution on strings (i.e., the set of strings with a certain minimum probability) from lossy or noisy samples. While the previous work concentrated on finding both the set of most probable elements and their probabilities, we consider *enumeration*, the problem of just finding a list that includes all the most probable elements without associated probabilities. Unlike Wigderson and Yehudayoff [10], we do not assume the underlying distribution has small support size, and our time bounds are independent of the support size. For the enumeration problem, we give a polynomial time algorithm for the lossy sample model for any constant erasure probability $\mu < 1$, and a quasi-polynomial algorithm for the noisy sample model for any *noise* probability $\nu < 1/2$ of flipping bits. We extend the lower bound for the number of samples required for the reconstruction problem from [3] to the enumeration problem to show that when $\mu = 1 - o(1)$, no polynomial time algorithm exists.

Keywords: Population Recovery, Enumeration of Heavy Hitters, Learning Discrete Distributions.

1 Introduction

Say that you are an investigator investigating DNA evidence at a crime scene. You can collect and analyze random DNA strands available at the scene, and you want to find DNA from each person involved in the crime, whether perpetrator or victim. There are several complications that make your task more difficult. First, it is not possible to exhaustively search through the huge number of microscopic strands of DNA at the scene; the best you can do is randomly pick strands and sequence them. Secondly, much of the DNA might have nothing to do with the crime. There might be small amounts of trace DNA from a huge number of people who passed by the scene at some time before the crime took place,

* Work supported by the Simons Foundation, the Ellentuck Fund, the Friends of the Institute for Advanced Study, and NSF grants DMS-0835373, CCF-121351 and CCF-0832797 subcontract No. 00001583.

** This research is supported by NSF grant CCF-1213151 from the Division of Computing and Communication Foundations.

P. Raghavendra et al. (Eds.): APPROX/RANDOM 2013, LNCS 8096, pp. 347–362, 2013.

and there might be contamination later. Thirdly, even the DNA from the people involved will be only partially recoverable, with either missing pieces or random noise.

Ideally, a complete crime scene analysis would give not just which DNA sequences were found, but their relative proportions. However, it would also be extremely useful to know the set of sequences present. You couldn't hope to get a complete list of trace sequences without sequencing every single strand. But you would want to filter out trace elements anyway, to concentrate on those from likely suspects.

This example illustrates the general issue of trying to analyze a distribution from lossy or noisy samples. Such problems are central to statistics, and arise in a variety of scientific circumstances. In theoretical computer science, Kearns et al. [6] introduced the general question of when a distribution can be identified from samples, and gave the first algorithms for formulations of the problem above. After that, attention was mainly focused on the continuous version of learning mixtures of Gaussians, which had been introduced by the statistician Pearson in the nineteenth century and was already a subject of great interest in AI and statistics. This problem spurred some highly interesting and deep algorithmic work (for example, [2,9,8,1]).

The equally fascinating case of lossy or noisy discrete distributions, e.g., lossy or noisy distributions on strings, only started getting attention again relatively recently. In particular, Dvir et al. [3] used an algorithm to infer a distribution on strings from lossy samples as a sub procedure in a learning algorithm for DNFs in the restriction model. Their goal was to learn the underlying distribution by giving an explicit description of a distribution which is close to the distribution that the samples were drawn from. For this reason, the problem as they formalized it assumed that the distribution had small support, with at most k non-zero probability elements, for a known parameter k which could affect the running time of the algorithm. However, none of the techniques they used relied on this assumption. On the other hand, the quasi-polynomial recovery algorithm of Wigderson and Yehudayoff [10] from noisy samples does rely on the fact that the support size is small.

In this work, we introduce a goal that is less ambitious but potentially more robust. There are many situations, like the crime scene investigation mentioned above, when it is desirable to identify a set containing all the probable elements, but it is not necessary to provide a complete description of the distribution. For example, perhaps one wants to identify gene patterns that are relatively common among the DNA of drug-resistant bacteria, texts that are most duplicated on web pages, or snippets of code that appear in many computer viruses. This problem of identifying the "heavy hitters" of a distribution, has been extensively studied in the context of streaming algorithms, see, for example, [5]. While just identifying the heavy hitters requires less information than exactly characterizing the distribution, for the same reason, it makes sense even when the distribution has no nice description, e.g., when the distribution has a constant fraction of mass divided among a very large, arbitrary set of strings. In this sense, algorithms for

heavy hitters can be *agnostic* in the sense of agnostic learning. We wish to find the best fit to a distribution with small support without relying on the assumption that the distribution actually has small support.

We consider two basic error models. In the lossy sample model, each bit of the sample drawn from the distribution is independently erased with probability μ. For the lossy model, we provide a polynomial time algorithm that identifies heavy hitters for any constant erasure probability $\mu < 1$. The error models for which our algorithm works are very general, and we do not require independence of noise in a bit from either the noise in other bits or the input, as long as the chance of erasure is never too large. For the case of independent erasures, the estimation algorithms of [3,7] can also solve the enumeration problem. As a converse, we also show that, if the erasure probability is $1 - o(1)$, no polynomial time algorithm exists for enumerating the heavy hitters.

In the noisy sample model, each bit of the sample drawn from the distribution is independently flipped with probability ν. For the noisy sample model, we give a quasi-polynomial time algorithm for identifying heavy hitters, for any constant $\nu < 1/2$. This algorithm is related but incomparable to the quasi-polynomial distribution recovery algorithm of Dvir et al. [10]. On the one hand, our algorithm is considerably faster (our exponent is doubly logarithmic rather than logarithmic), and does not require the distribution to have small support. On the other hand, they solve the harder problem of estimating the probabilities for each heavy hitter.

1.1 Problem Definitions and Formal Statements of Our Results

The general issue of learning about distributions from noisy samples has many variations and interesting formulations. Below, we will consider the ingredients of such formulations and how they relate to each other.

Underlying distribution. In all versions, the samples come from some underlying distribution D on $\{0,1\}^n$. D is unknown to the algorithm. The algorithm can only access samples drawn from the D after errors are introduced. As mentioned earlier, [3,10] make the assumption that D has support size at most k, where k is given to the algorithm. Run times are given in terms of k and n. We call this the *small support size* case. We mainly consider the case when D is *arbitrary*. One could also consider other families of distributions, e.g., easily sampled distributions, high-entropy distributions or distributions that consist of independent pairs of strings x, y.

Error models. Like [3,10], we examine two basic types of error. The basic *lossy error* model is as follows. After a string $x = x_1 \dots x_n$ is drawn from D, the observed sample has the form $y_1 \dots y_n$ where for each i independently, we set $y_i = *$ with probability μ, and $y_i = x_i$ otherwise, where $*$ is a new symbol. The difficulty of the problem in the lossy model depends on the constant μ, the larger the μ the more difficult the problem is. Some of our algorithms and one of the algorithms in Dvir et al. [3] only work for μ smaller than a specified constant.

In the *noisy* model, the observed sample has the form $y_1 \dots y_n$ where for each i independently, we set $y_i = 1 - x_i$ with probability ν, and $y_i = x_i$ otherwise.

We can also consider the problem for other error models. For example, our lossy enumeration algorithm works equally well in a semi-random erasure model, where, after erasures occur, an adversary is allowed to "un-erase" an arbitrary set of positions. It is easy to see that estimation is not possible in such a model.

Goal. We distinguish three goals for an algorithm. We are primarily concerned with the *enumeration* problem. Here, the algorithm has as input a parameter $\epsilon > 0$, and needs to output a list L that contains all ϵ-heavy hitters of D, i.e., all the strings z so that $\mathbf{P}[x = z] \geq \epsilon$. L may contain some non-heavy hitters, but we expect the algorithm to explicitly list the elements of L, so an efficient algorithm cannot output a large list. There may in general be $1/\epsilon$ heavy hitters, so an ideal algorithm runs in polynomial time in n and $1/\epsilon$.

In the *estimation* problem, the algorithm is given z and ϵ, and is required to output an estimate of $\mathbf{P}[x = z]$ that is correct within additive error ϵ. Even without errors, getting such an estimate would require $1/\epsilon^2$ samples, so again polynomial time in n and $1/\epsilon$ is the best we could hope for.

Finally, in the *recovery* problem, we wish to find a list of elements that contains all 2ϵ-heavy hitters and only contains ϵ-heavy hitters, and for each element on our list we wish to have an estimate of its probability that is within an additive error of at most ϵ. Note that, for distributions with support k, a recovery algorithm with $\epsilon = \delta/k$ will provide an explicit description of a distribution that is within δ of the actual one. This is the sense that a recovery algorithm actually "recovers" the original distribution.

The recovery problem combines the enumeration and estimation problems, but Dvir et al. [3] observe that any estimation algorithm can be used to solve the recovery problem with only a factor n overhead, via a "branch-and-prune" approach. Thus, we only need to actually look at enumeration and estimation, and estimation is the more difficult problem.

We devise or analyze algorithms for several of these problems. The results we present in this extended abstract are:

1. For any constant $\mu < 1$, we give a polynomial time (in n and $1/\epsilon$) algorithm for enumeration of the ϵ-heavy hitters from lossy samples where μ is the bit erasure probability. Here, the distribution is arbitrary and the time does not depend on the support size of the distribution. The list size is a polynomial in $1/\epsilon$ that does not depend on n.

   In fact, we need only the following property of the erasure model: There is a parameter T so that, for any subset of T bit positions, the probability that all bits are erased is $o(\epsilon^2)$, whereas the probability that no bit is erased is at least $poly(\epsilon)$. For independent erasures, both are true when $T = C \log 1/\epsilon$, where C is a constant depending on μ. But this property will also be inherited by any samples with fewer erasures, such as the semi-random distribution

mentioned above. It will also hold for e.g., T-wise independent distributions on erasures.
2. For samples with independent noise ν for any constant $\nu < 1/2$, we give an enumeration algorithm that takes polynomial time in n, and quasi-polynomial time in $1/\epsilon$ (more precisely, time $\text{poly}(n)\epsilon^{O(\log\log\epsilon)}$) time . Here, the distribution is arbitrary and the time does not depend on the support size of the distribution. The list size does not depend on n.
3. [3] give a super-polynomial lower bound for estimation from lossy samples even for small support distributions when $\mu = 1 - o(1)$. We give a similar lower bound for enumeration from lossy samples for small support distributions.
4. Dvir et al. [3] also observe that LP-duality can be used to characterize the sample complexity of algorithms for estimation for arbitrary distributions, for any error model. We give a relaxation of this LP program that shows a "Yao principle" for such estimation problems. We show that either there is an algorithm for estimation using a certain number of samples or a polynomially-related lower bound via two distributions whose noisy versions are indistinguishable. [3] also introduce the notion of "local inverse" to a matrix. We show that the existence of any algorithm for estimation for arbitrary distributions in an error model implies one via a local inverse to the corresponding matrix.

In the full paper, we will include some additional results, not directly related to enumeration:

1. We give a new algorithm for estimation from lossy samples that works in time polynomial in n and exponential in $1/\epsilon$ whenever $\mu \leq 2/3$.
2. Dvir et al. [3] have shown that their estimation algorithm for lossy samples will work for arbitrary distributions for $\mu < 0.614....$ We give a tighter analysis of their algorithm, showing that it works when $\mu < 1 - 1/\sqrt{2} = 0.69....$ We present numerical computations that suggest that the real threshold for this algorithm is $\mu = .75$.

1.2 State-of-the-Art

For lossy samples with independent erasures, the current best algorithms for all versions of the problem are due to Moitra and Saks [7], who, subsequently to our work, solve the hardest version of this problem. They give a polynomial time algorithm for estimation for arbitrary distributions, which implies similar algorithms for enumeration and recovery. This result improves on the estimation and recovery algorithms of [3], and is better than the algorithms from the first two of our additional results to appear in the full paper. Our enumeration algorithm is still useful if the erasures are not actually independent.

For noisy samples, for distributions of small support size, the current best algorithm for estimation and recovery problems is the quasi-polynomial algorithm of [10]. For any constant level of noise, this algorithm runs in time polynomial in n and $k^{\log k}$ when the support size is k. Our second result above is the only

non-trivial algorithm known algorithm for enumeration when the distribution is arbitrary, and is the fastest algorithm for enumeration even when the support is small. For the estimation and recovery problems when the distributions are arbitrary, no algorithms better than exponential time in n are known.

Combining the lower bound of [3] and the lower bound in result three above, all versions of these problems require more than polynomially many samples when the bit erasure probability $\mu = 1 - o(1)$ or when the bit flipping probability is $\nu = 1/2 - o(1)$.

2 Branch-and-Prune Algorithms

Our algorithms for enumeration follow the same "branch-and-prune" paradigm as those of [3,10]. This paradigm can be viewed as a form of classical dynamic programming [11], but in the context of probabilistic algorithms for enumeration, the first example we know of is the Goldreich-Levin algorithm for list decoding the Hadamard code. [4]. Dvir et al. [3] used a branch-and-prune method to reduce the recovery problem to the estimation problem. In this section, we revisit this connection and present a self-contained explanation. We also emphasize the minimal set of conditions that a pruning algorithm needs to have to be useful for enumeration.

Let D be the underlying distribution on n bit strings. For $1 \leq m \leq n$, let D_m be the distribution on the first m bits of a string drawn from D. In the context of enumerating heavy hitters, we observe that the distribution on the first m bits of the observed sample is equal to the distribution on lossy/noisy samples from D_m with the same error parameter μ or ν. Also note that the m-bit prefix of any heavy hitter for D is a heavy hitter for D_m. Our goal is, for m from 1 to n, to find a set S_m of candidates that contains all the heavy hitters of D_m. Then the heavy hitters for D_{m+1} are contained in the set $T_{m+1} = \{x0 | x \in S_m\} \cup \{x1 | x \in S_m\}$. This is the "branch" stage. However, to prevent this set of candidates from growing exponentially, it is necessary to "prune" this set back to a smaller subset S_{m+1}.

To be more precise, a *pruning algorithm* takes a set of m-bit strings T and can request lossy or noisy samples from D_m, and produces a subset $S \subset T$. We say the strings in $T - S$ are *pruned* by the algorithm. Our pruning algorithms have a parameter s and we need two properties to ensure their efficiency:

Correctness. With high probability $(1 - o(1/(ns)))$, no heavy hitter for D_m is pruned.

Efficiency. If $|T| > s$, then with high probability S is a strict subset of T.

Note that an estimation algorithm can be used as a pruning algorithm: Estimate the probability of each element of T to within an additive term of $\epsilon/3$. Prune the ones whose estimates are less than $2/3\epsilon$. Assuming the estimates are correct, all ϵ heavy hitters are maintained. Furthermore, all but at most $s = 3/\epsilon$ elements with probability at least $\epsilon/3$ are pruned.

If both of these conditions hold, we can use the pruning algorithm for enumeration. We maintain a set S_m that with high probability contains all heavy hitters

for D_m. We construct T_{m+1} as above, and prune it until the size remaining is at most s. Since T_{m+1} has at most $2s$ elements at the start, and each time we prune we get a strict subset, pruning can happen at most s times for any m. The probability of pruning a heavy hitter in any one run of the pruning algorithm is $o(1/(sn))$ so with high probability, we never prune a heavy hitter. We will need to use the pruning algorithm $O(ns)$ times on a set of strings of size $O(s)$.

If we have an estimation algorithm and an enumeration algorithm, we can enumerate a list S_n of heavy hitters and estimate the probability of each of the candidate heavy hitters in S_n to get a recovery algorithm.

Thus, it suffices to describe an algorithm for estimation or pruning in order to specify an algorithm for recovery or enumeration, respectively.

3 Enumeration Algorithms

In this section, we give a polynomial time enumeration algorithm for lossy samples for any $\mu < 1$, and a quasi-polynomial time algorithm for noisy samples for any $\nu < 1/2$. As described before, we only need to give a pruning algorithm in each case.

3.1 Lossy Samples

We describe here the pruning algorithm for lossy samples. Let $\epsilon > 0$. For some polynomial $s = n^{O(1)}\epsilon^{-O(1)}$, we are given a set T of at most $2s$ strings that includes all ϵ-heavy hitters, and want to find a subset S that contains all ϵ-heavy hitters, but has at most s strings. When the parameter ϵ is clear from the context, we will simply refer to an ϵ-heavy hitter as a heavy hitter.

The pruning algorithm works in two phases. In the Phase I, we will collect a maximal set $B \subseteq T$ of *centers* of small size such that the centers are at a distance of at least $d = C \log \frac{4}{\epsilon}$ from each other, where C is a constant which only depends on μ and will be determined later. Since B is maximal, each heavy hitter is either in B or at most at a distance of d from an element in B. In the second phase, for each center, we consider the elements of T close to it and prune them.

Phase I: In Phase I, we start with a set B of centers (initially an empty set) of size no more than $2/\epsilon$ and greedily place non-pruned strings from T into B as long as $|B| \leq 4/\epsilon$ so that all centers in B are at least a distance d from each other. If the size of B never equals $\lceil 4/\epsilon \rceil$, we proceed to Phase II. If not, we execute the following sub procedure to prune the non-heavy hitters from B so that the size to B reduces to no more than $2/\epsilon$. We then repeat the greedy of process of growing B.

Pruning B: At this point, we have $|B| = 4/\epsilon$. Our goal is to cut the size of B by half by pruning away sufficiently many non-heavy centers of B.

For $v, w \in T$ define $\Delta_{v,w} = \{i | v_i \neq w_i\}$. In other words, $\Delta_{v,w}$ is the set of positions where the strings v and w differ. For $u, u' \in B$ and for a lossy sample

y, we say that y is (u, u')-*discriminatory* if for at least one position in $\Delta_{u,u'}$, the value is not erased in y. We say y is *discriminatory* if it is (u, u')-discriminatory for all $u, u' \in B$. y is *consistent* with a string $u \in T$ if it never disagrees with u in a revealed bit. Note that y is always consistent with the original sample.

To prune non-heavy strings in B, we draw $t = \text{poly}(1/\epsilon)$ lossy samples. For each center u, we compute the fraction p_u of samples that are discriminatory and consistent with u. Each sample can contribute to at most one center since a discriminatory sample can be consistent with at most one center. If a discriminatory sample y were to be consistent with centers u and u', then there is a position in $\Delta_{u,u'}$ which is not erased in y where u and u' agree, a contradiction. Finally, if $p_u < \epsilon/2$, we prune u.

We will argue that at least $2/\epsilon$ strings of B are pruned by the procedure so we will end up with a B of size at most $2/\epsilon$. Let y be a lossy sample. y is discriminatory with probability at least $7\epsilon/8$ since

$$\begin{aligned}\mathbf{P}[y \text{ is not discriminatory}] &\leq \sum_{u,u' \in B} \mathbf{P}[y \text{ is not } (u,u')\text{-discriminatory}] \\ &\leq (8/\epsilon^2)\mu^d \leq (8/\epsilon^2)\mu^{C \log 4/\epsilon} \\ &= (8/\epsilon^2)(\epsilon^3/64) \text{ for } C = 3/\log \tfrac{1}{\mu} \\ &\leq \epsilon/8\end{aligned}$$

Let u be a ϵ-heavy hitter. The probability that y is discriminatory and is consistent with u is at least $7\epsilon/8$. Let p_u be the fraction of discriminatory samples consistent with u. $\mathbf{P}[p_u \leq \epsilon/2] \leq e^{-2(3\epsilon/8)^2 t}$ by Chernoff-Hoeffding bound since the expected fraction is at least $7\epsilon/8$. $\mathbf{P}[\exists$ an ϵ-heavy center u such that $p_u \leq \epsilon/2] \leq \frac{1}{\epsilon} e^{-2(3\epsilon/8)^2 t}$ by union bound since there are at most $1/\epsilon$ such centers. Since $t = \text{poly}(1/\epsilon)$, and since there are at most $2/\epsilon$ centers that pass the threshold of $\epsilon/2$, with high probability, we will have pruned at least $2/\epsilon$ centers while retaining all heavy hitters in B.

Phase II: At this point, we have a set B of centers of size at most $4/\epsilon$ such that every non-pruned element of T is within a distance of d from some center. Assign each non-pruned element of T to its closest center. For $u \in B$, let B_u denote the set of elements of T assigned to u.

For each u, we prune B_u so that its size is bounded by a polynomial in $1/\epsilon$. Since there are at most $4/\epsilon$ centers, we will end up with the desired bound after pruning. Fix $u \in B_u$. In the following, we outline how to prune B_u.

For $v \in B_u$, let Δ_v denote the set of positions where u and v differ. We say that a sample y is *revelatory* for v if for no position i in Δ_v, $y_i = *$.

We draw t lossy samples. For each v, we compute the fraction of samples that are revelatory for and consistent with v. Each sample can contribute to at most one element of B_u since a sample y can be revelatory for and consistent with at most one $v \in B_u$. For every $v, v' \in B_u$ and $v \neq v'$, there exists a position in $\Delta_v \cup \Delta_{v'}$ where one of v and v' agrees with u and the other disagrees with u. If y were to be revelatory for and consistent with both v and v', then v and v'

agree on every position in $\Delta_v \cup \Delta_{v'}$, which is a contradiction. If the fraction for v does not exceed the target $\epsilon^{C'+1}/2$, we prune it, where C' is a constant that depends only on μ.

Let $v \in B_u$ and y be a sample. We will show that the probability a sample y is revelatory for v is not small.

$$\begin{aligned}\mathbf{P}[y \text{ is revelatory for } v] &\geq (1-\mu)^{|\Delta_v|} \geq (1-\mu)^d = (1-\mu)^{3\log\frac{4}{\epsilon}/\log\frac{1}{\mu}} \\ &= (\epsilon/4)^{C'} \text{ for some constant } C' \text{ which only depends on } \mu\end{aligned}$$

Let $v \in B_u$ be an ϵ-heavy hitter. The probability that y is revelatory for and consistent with v is at least $(\epsilon/4)^{C'+1}$. Let p_v be the fraction of samples that are revelatory for and consistent with v. $\mathbf{P}[p_v \leq (\epsilon/4)^{C'+1}/2] \leq e^{-2((\epsilon/4)^{C'+1}/2)^2 t}$ by Chernoff-Hoeffding bound since the expected fraction is at least $(\epsilon/4)^{C'+1}$. $\mathbf{P}[\exists$ an ϵ-heavy v such that $p_v \leq (\epsilon/4)^{C'+1}/2] \leq \frac{1}{\epsilon} e^{-2((\epsilon/4)^{C'+1}/2)^2 t}$ by union bound since there are at most $1/\epsilon$ such elements. Since $t = \text{poly}(1/\epsilon)$, with high probability, at most $2/(\epsilon/4)^{C'+1}$ elements remain after pruning since there are at most $2/(\epsilon/4)^{C'+1}$ elements $v \in B_u$ such that $p_v \geq (\epsilon/4)^{C'+1}/2$.

A General Lossy Error Model: Our algorithm for enumerating ϵ-heavy hitters in the lossy error model and its analysis does not require that each bit position is erased independently with probability μ. Our algorithm and its analysis can be easily extended to lossy error models that satisfy weaker conditions.

Let x be an arbitrary sample drawn according to the original distribution D. Let y be a lossy model obtained from x according to a lossy error model. In the following we provide a sufficient condition for lossy error models that guarantees the correctness and performance of the above algorithm.

For all $\epsilon > 0$, there exists a t such that for all sets S of t positions,

1. $\mathbf{P}[$values in all positions of S are erased$|x] \leq \epsilon^3/64$, and
2. $\mathbf{P}[$none of the values in positions of S are erased$|x] \geq \epsilon^{O(1)}$.

3.2 Noisy Samples

Here we give a quasi-polynomial enumeration algorithm for the noisy case for any constant $0 \leq \nu < 1/2$. As before, we present just the pruning algorithm. As in the lossy case, it works in two phases. First, we locate a small number of *centers* so that every heavy hitter is $C \log 1/\epsilon$ distance from one of the centers, where C is a constant that only depends on μ. This already gives a $s = n^{O(\log 1/\epsilon))}$ algorithm, which works under a very general noise condition. However, we can improve it to $s = (1/\epsilon)^{O(\log(\log(1/\epsilon)))}$ in the second stage that requires noise to be exact and independent. In the second stage, for each center, we enumerate the heavy hitters within $d = O(\log 1/\epsilon)$ Hamming distance to the center. For each fixed center, this is equivalent to enumerating all of the low Hamming weight heavy hitters, which we can identify with the set of positions with value 1. To do this, we give a potential function for a set of positions A, which upper bounds the total probability of small Hamming weight strings that could contain A.

In addition to A, we'll maintain a set of positions R so that we only need to look at heavy hitters whose 1's are in R. We show that we can divide the extensions $A \cup \{i\}$ into two categories, one that significantly lowers the potential function, and another that significantly reduces the set R of positions we need to consider in the future. If the potential function drops below ϵ, we can prune the search. If R becomes very small, we can use a brute force search on small subsets of R.

The first phase works very similarly to the erasure case. We start with a set B of centers (initially an empty set) of size no more than $2/\epsilon$ and greedily place non-pruned strings from T into B as long as $|B| \leq 4/\epsilon$ so that all centers in B are at least a distance d from each other. If the size of B never equals $\lceil 4/\epsilon \rceil$, we proceed to phase two. If not, we execute the following sub procedure to prune the non-heavy hitters from B so that the size to B reduces to no more than $2/\epsilon$. We then repeat the greedy of process of growing B.

For $u, v \in B$, let $\Delta_{i,j}$ be the set of positions where u and v differ. We say that a noisy sample y *favors* $u \in B$ if for every $v \in B$, y agrees with u in strictly more than half the positions in $\Delta_{u,v}$. Note that, if u is itself is the original sample, each bit of the noisy sample y agrees with that of u with probability $1 - \nu > 1/2$. So the probability that at least half the positions of y in $\Delta_{u,v}$ disagree with u, by Chernoff-Hoeffding bound, is at most $e^{-\Omega(d(1/2-\nu)^2)} < \epsilon/16$ for some $C = O(1/(1/2 - \nu)^2)$. So by a union bound over the $4/\epsilon$ centers, if the original sample is u, the conditional probability that y does not favor u is at most $1/4$. (This is the only place where we use the bound on the noise in the first phase.) Thus, for any ϵ-heavy hitter u, y will favor u with probability at least $3\epsilon/4$. Also note that y can favor at most one center u, because otherwise y would have to agree with both u and u in more than half the places where they differ. So we estimate, using $O(1/\epsilon^2 \log s/\epsilon)$ samples, the fraction of y's that favor each center, and prune all but the $2/\epsilon$ centers where this estimate is greater than $\epsilon/2$.

At this point, B, the set of centers, has size less than $4/\epsilon$, where every unpruned string of T is within Hamming distance d of one of the centers. Note that there can be at most $O(n^d/\epsilon) = n^{O(\log 1/\epsilon)}$ candidates left at this point, so if noise is not independent, we can simply use this phase as our pruning algorithm to get a quasi-polynomial time enumeration algorithm.

In phase two, for each center, we enumerate those heavy hitters that are within d to that center. The union of these lists will include all heavy hitters. By taking the bit-wise parity of the noisy samples with the center in question, we can without loss of generality assume that the center in question is the all-zero string. Thus, the problem is now equivalent to enumerating all heavy hitters of small Hamming weight, i.e., d or less. For a string x, let Δ_x be the set of positions where x is 1.

Our procedure is recursive. At each point, we have two sets of positions $A, |A| \leq d$ and R, and we are trying to enumerate all low Hamming weight heavy hitters x with $A \subseteq \Delta_x \subseteq A \cup R$. Initially, A is empty, and R will be all n positions (but we show that very quickly, R will decrease to just poly$(1/\epsilon)$ positions.) We use a potential function $0 \leq Q_{A,R} \leq 1$ that allows us to prune some branches that cannot lead to actual heavy hitters. A large value of

potential function is necessary but not sufficient for there to be such a heavy hitter x with $A \subseteq \Delta_x \subseteq R \cup A$. $Q_{A,R}$ has the following properties:

1. If $|A| \leq d$, $Q_{A,R}$ can be approximated to within any poly(ϵ) additive error in time poly($1/\epsilon$).
2. If there is an ϵ-heavy hitter x of Hamming weight $\leq d$ with $A \subseteq \Delta_x \subseteq A \cup R$, then $Q_{A,R} \geq \epsilon/2$.
3. If $|R| > O(\log^2 1/\epsilon)$ and $Q_{A,R} \geq \epsilon/2$, then the average value of $Q_{A\cup\{i\},R}$ for $i \in R - A$ is at most $|R|^{-1/4} Q_{A,R}$.

Property 1 allows us to compute the potential function. Property 2 allows us to prune any branch where the potential function is too small. Property 3 says that the potential function decreases dramatically for most cases where we extend A. We first define $Q_{A,R}$ and prove it has the above properties, then describe the enumeration algorithm in terms of the properties. In the following, we use x to refer to the original sample and y to the corresponding noisy sample.

Let a and b be constants (depending on ν) so that $a\nu + b(1-\nu) = 0$, and $a(1-\nu) + b\nu = 1$ (equivalently $a = (1-\nu)/(1-2\nu)$ and $b = -\nu/(1-2\nu)$). For bit position i, let v_i be a if $y_i = 1$ and b if $y_i = 0$. Then the equations above say that the conditional expectation of v_i is x_i. (This method is borrowed from the unbiased sampler for the estimation problem from [3]. Unfortunately, using that estimator takes exponential time. We get around this by only using it within A, which has size at most $d = O(\log 1/\epsilon)$.) Let w_{R-A} be the number of 1's in y within $R-A$. Let $E_{A,R}$ be an indicator variable for the event that $w_{R-A} \leq \nu|R-A|+d-|A|$. Let $V_{A,R}$ be the random variable $(\Pi_{i\in A} v_i)E_{A,R}$. Note that, for a fixed x, all of the factors in $V_{A,R}$ are independent, so $\mathbf{E}[V_{A,R}|x] = \mathbf{P}[E_{A,R}|x]\Pi_{i\in A} x_i$. (This equation uses the fact that the noise in each bit is independent.) Let $Q_{A,R}$ be the expectation of $V_{A,R}$. The above shows that although $V_{A,R}$ may be a polynomially large positive or negative number, $Q_{A,R}$ is always between 0 and 1.

The absolute values of the v_i's are bounded by a constant, and we multiply at most d of them, so the maximum value of V is polynomial in $1/\epsilon$. Thus, we can estimate $Q_{A,R}$ up to any polynomial in ϵ additive error by averaging $V_{A,R}$ for $poly(1/\epsilon)$ samples. This establishes the first property.

If x has Hamming weight at most d, and $A \subseteq \Delta_x \subseteq R \cup A$, then if the expected number or fewer of bit flips occur in $R - A$, $E_{A,R}$ will be true. Thus, for such an x the conditional probability of $E_{A,R}$ is at least $1/2$. For such an x, $\mathbf{E}[V_{A,R}|x] \geq 1/2\Pi_{i\in A} x_i = 1/2$. Since the conditional expectation is never negative, if there is any ϵ-heavy hitter as above, $Q_{A,R} \geq \epsilon/2$. This establishes the second property.

To establish the third property, let $h_0 = (d - |A|)/(1-2\nu) + 4\sqrt{|R| \log 1/\epsilon}$. Fix any x, and let h be the Hamming weight of x in $R - A$. We claim that $\sum_{j\in R-A} \mathbf{E}[V_{A\cup\{j\},R}|x] \leq h_0(\mathbf{E}[V_{A,R}|x] + \epsilon^2)$. Note that for any $j \in R - A$, $E_{A\cup\{j\},R}$ implies $E_{A,R}$. Then

$\mathbf{E}[V_{A\cup\{j\},R}|x] = x_j(\Pi_{i\in A} x_i)\mathbf{P}[E_{A\cup\{j\},R}|x] \leq x_j(\Pi_{i\in A} x_i)\mathbf{P}[E_{A,R}|x] = x_j\mathbf{E}[V_{A,R}|x]$. Summing all $j \in |R-A|$, we get an upper bound of $h\mathbf{E}[V_{A,R}|x]$ for

the conditional expectation of the sum. So the inequality holds conditioned on any x with $h \leq h_0$.

For the other case, fix x with $h = fh_0$, $f \geq 1$. Then the expected Hamming weight of y within R is $\nu|R|+h(1-2\nu)$, and our cut-off for $E_{A,R}$ to occur is $\nu|R|+(d-|A|)$. Applying Chernoff bounds, we can upper bound the probability of $E_{A,R}$ given such an x as at most ϵ^{2f^2}. Then the overall expectation of $hE[V_{A,R}|x] \leq Ah_0\epsilon^{2f^2}$, which is decreasing with f. So setting $f = 1$ gives us an upper bound which holds in general, of $h_0\epsilon^2$ which is the second error term.

By linearity of expectation, then $\sum_j Q_{A\cup\{j\},R} \leq h_0(Q_{A,R} + \epsilon^2) \leq 2h_0Q_{A,R}$ since $Q_{A,R} \leq \epsilon/2$. So this is at most $O(\sqrt{|R-A|\log 1/\epsilon})Q_{A,R}$. Dividing by $|R-A|$ and using the assumptions $|R| \geq C'(\log 1/\epsilon)^2$ for some sufficiently large constant C' gives us that the average value of $Q_{A\cup\{j\},R}$ is at most $Q_{A,R}R^{-1/4}$. This establishes the last property.

We use this potential function $Q_{A,R}$ in our algorithm as follows: At any point, we will be enumerating those heavy hitters x with Hamming weight at most d so that $A \subset \Delta_x \subset R$ for some sets $A, |A| \leq d$ and R.

If $R < C' \log^2 1/\epsilon$, we just enumerate all d-tuples by brute force. If $Q_{A,R} < \epsilon/2$, we can just terminate and return the empty set.

Otherwise, we compute $Q_{A\cup\{j\},R}$ for each $j \in R - A$. Divide those j into the "exceptional" ones with $Q_{A\cup\{i\},R} \geq |R|^{-1/8}Q_{A,R}$ and the remaining "typical" ones. By the third property and Markov's inequality, at most $O(|R|^{7/8})$ can be exceptional. Let R' be the set of exceptional positions.

For any heavy hitter x with $A \subseteq \Delta_x \subset R$, either $\Delta_x - A \subseteq R'$ or $j \in \Delta_x$ for some $j \in R - R'$. So we output A to the list of heavy hitters, then recurse on (A, R') and recurse on $(A \cup j, R)$ for each $j \in R - R'$. This covers all of the above cases recursively.

We now have to give a time analysis for the above algorithm. Let $K = \lfloor 2Q_{A,R}/\epsilon \rfloor$ be a measure of how far we are from being able to prune our current set. Let r represent the size of R. We give our bound in terms of the number of recursive calls $T(K, r)$ needed for these values. First, if $K = 0$, we prune and terminate. So $T(0, r) = 1$. If $r < O(\log^2 1/\epsilon)$, we use brute force search and take time $(1/\epsilon)^{O(\log\log 1/\epsilon)}$. If $r > O(1/\epsilon^8)$, we make only one recursive call that isn't immediately pruned, to a subset R' of size at most $O(|R|^{7/8})$. So we shrink R without branching until $r = O(1/\epsilon^8)$. Otherwise, we make up to r "typical j" recursive calls where K decreases by a $r^{-1/8}$ factor, and one "extraordinary" recursive call where r becomes $O(r^{7/8})$. Thus, we have the recursion $T(K, r) \leq rT(Kr^{-1/8}, r) + T(2/\epsilon, r^{7/8})$. Unwinding the first part of the recursion, each time we lose a factor of $r^{1/8}$ from K, it costs us a factor of r. So we get $T(K, r) \leq K^8T(2/\epsilon, r^{7/8})$. This recursion has $O(\log\log r) = O(\log\log 1/\epsilon)$ depth, each giving a factor of $(2/\epsilon)^8$ for a total of $(1/\epsilon)^{O(\log\log 1/\epsilon)}$ recursive calls. The leaves of the recursion cost us a similar factor as mentioned earlier. During each recursion, we have to compute Q r times, which gives another polynomial factor to the total running time. Thus the total time is $(1/\epsilon)^{O(\log\log 1/\epsilon)}$, as is the list length.

4 Lower Bounds on Enumeration

We first review the result from [3] on the lower bound for the number of samples required for estimation. Let $n = \log 1/\alpha$. Consider two distributions D_0 and D_1. D_1 is the uniform distribution over n-bit strings, D_0 the uniform distribution on those n-bit strings with even parity. In D_0 the all-zero string has probability 2α, and in D_1, α, so any $\alpha/3$ estimation algorithm has to distinguish between the two.

However, both distributions are uniform when we restrict strings to any proper subset of the bit positions. Let $Q = (Q_1, Q_2, \cdots, Q_t)$ be any sequence where for $1 \leq i \leq t$, $Q_i \subset [1..n]$ is a set of bit positions. Assume that for each $1 \leq i \leq t$, i'th sample reveals exactly the bits in positions given by Q_i. Based on this condition and assuming that none of the Q_i is the entire set of positions, the induced distributions on lossy samples are identical. Any distinguishing algorithm has to wait until it sees at least one sample with all bits revealed. The probability of this occurring with any one sample is $(1-\mu)^n$, so the time required is at least $(1-\mu)^{-n} = (1-\mu)^{\log \alpha} = \alpha^{\log(1-\mu)}$. Hence if $\mu = 1 - o(1)$, this is $(1/\alpha)^{\omega(1)}$ and is super-polynomial.

Note that these two distributions give no lower bound on the number of samples required for enumeration, since an enumeration algorithm could just list all $1/\alpha$ strings of length n without seeing any samples at all. However, we will present a very similar argument that shows a lower bound for enumeration. For this, we need to increase the value of n without increasing the support size. To achieve this condition, we look at sparse distributions. We will use the fact that we are unlikely to reveal even a small constant fraction of the bit positions when $\mu = 1 - o(1)$. The following lemma and its corollary establish the existence of the desired distribution, which are stated without proof.

Lemma 1. *Let $1/2 > \gamma > 0$. There is a γn-wise independent distribution on strings of length n that has support size $2^{O(\gamma \log(\frac{1}{\gamma})n)}$.*

Corollary 1. *For any string x of length n, there is a distribution on strings of length n that is γn-wise independent, and where x has probability at least $2^{-O(\gamma \log(\frac{1}{\gamma})n)}$.*

Now, let $\mu = 1 - 1/L$. If $\mu = 1 - o(1), L = \omega(1)$. We will show that the sample complexity or the list size of any algorithm the enumerates ϵ-heavy hitters from lossy samples with erasure probability μ is super-polynomial in $1/\epsilon$. Let G be such that $L = eG^{1+\frac{G}{\log G}}$ and let $\gamma = 1/G$. G will be $\Theta(\log L)$ (since the right side is $2^{\Theta(G)}$). For any n, select a random n-bit string x and run the enumeration algorithm with $\epsilon = 2^{-\Omega(\gamma n \log \frac{1}{\gamma})} = 2^{-O(\log G/G)}$ using the γn-wise independent distribution with x in its support as obtained from the corollary. Say that the algorithm gets t lossy samples and enumerates a list of size t'. Let $S = (S_1, \cdots, S_t)$ be the sequence of sets of revealed bit positions in the samples. If every set in the sequence S has size at most γn, the observed samples are independent of x. Hence the conditional probability that x is on the

list is at most $t'/2^n$. So either $t' = \Omega(2^n) = (\frac{1}{\epsilon})^{\Omega(G/\log G)} = (\frac{1}{\epsilon})^{\Omega(\log L/\log\log L)}$ or there is a constant probability that some $|S_i| > \gamma n$. Since each position is revealed with probability $1/L$ and there are $\binom{n}{\gamma n} \leq (e/\gamma)^{\gamma n} = (eG)^{n/G}$ subsets of γn positions, the probability that any one S_i is of size $\geq \gamma n$ is at most $(eG)^{n/G}(1/L)^{n/G} = (eG/L)^{n/G}$. Hence, for any one of the S_i to have size more than γn, we must have $t > (L/eG)^{n/G} = (G^{G/\log G})^{n/G}$ by our choice of G. Now, $\epsilon = 2^{-\Omega((n\log G)/G)} = G^{-\Omega(n/G)}$, so in this case we have $t = \epsilon^{-\Omega(G/\log G} = (\frac{1}{\epsilon})^{\Omega(\log L/\log\log L)}$. Thus, if L is $\omega(1)$, then either the list size or the number of samples must be super-polynomial in $1/\epsilon$, with exponents of the form $\Omega(\log L/\log\log L)$.

The quantitative bound we get here for enumeration is almost as good as the one above for estimation. As far as we know, these are the best known lower bounds for these problems. However, they are pretty far from the upper bounds.

5 Canonical Upper and Lower Bounds for Estimation

Here, we use LP duality to show that either there is a lower bound for estimation of a certain canonical form, or an algorithm of a canonical form.

The lower bound for estimation shown in [3] has the following form: there are two distributions that differed by at least α in their probability of 0^n, the all-zero string. However, the induced distributions on lossy samples were statistically close. The distributions in [3] were completely indistinguishable unless all bits were revealed, which can only happen with small probability. Consider for now the noisy case with bit-flipping probability of ν. Any distributions D_0, D_1 that differ by α in the all-zero string must be distinguishable by an $\alpha/3$ estimation algorithm. Let N_0, N_1 be the corresponding noisy distributions, and let λ be the statistical distance of N_0 and N_1. Then any algorithm that distinguishes the two requires $t = \Omega(1/\lambda)$ samples. Now, without loss of generality, we can assume that both distributions are symmetric, so that the probabilities of outputting a Hamming weight i string for $0 \leq i \leq n$ determine them. Let Δ_i be the difference of these two probabilities for weight i. Then $\Delta_0 > \alpha$, $\sum \Delta_i = 0$ and $\sum |\Delta_i| \leq 2$. Conversely, any values of Δ_i satisfying these inequalities give rise to two such distributions whose differences are Δ_i for $0 \leq i \leq n$.

Let $m_{i,j}$ be the probability that a noisy version of a sample of Hamming weight i ends up with Hamming weight j. An explicit formula for $m_{i,j}$ is $m_{i,j} = \sum_k \binom{i}{k}(1-\mu)^k \mu^{i-k}\binom{n-i}{j-k}\mu^{j-k}(1-\mu)^{n-i-j+k}$, but we will not be using particular noisy error model right now. Our treatment works more generally for any Hamming weight to Hamming weight transformation matrix. Then the difference between N_0 and N_1's probabilities of producing a Hamming weight j string is $\sum_i m_{i,j}\Delta_i$. Thus, if we include the inequality $\sum_j |\sum_i m_{i,j}\Delta_i| \leq \lambda$ in addition to the ones above, we get that the statistical distance of the noisy versions is at most λ. Thus, these linear inequalities characterize the existence of such distributions.

In fact, these inequalities make sense in a very general setting. Let M be a stochastic $n_1 \times n_2$ metric with entries $m_{i,j}$ representing the probability: if the

original sample is of type i, that the observed sample is of type j. The above inequalities say that there are two distributions on types so that $1/\lambda$ observed samples are required to distinguish them, but differ by α in the probability of type 0 in the original distributions. So, in particular, any $\alpha/3$ estimator of the probability of type 0 requires $1/\lambda$ samples. Call such a pair of distributions (α, λ)-*fooling* pair of distributions for M.

We now consider a somewhat simplified relation of the inequalities above, but which preserve the parameters to within polynomial factors. Fix a value for λ. The primal relaxation has objective to maximize Δ_0 subject to the following inequalities in variables $\Delta_0, \ldots, \Delta_n$: 1. $\sum_{i=0}^{i=n} \Delta_i = 0$, 2. For each $0 \leq i \leq n_1$, $-1 \leq \Delta_i \leq 1$, and 3. For each $0 \leq j \leq n_2, -\lambda \leq \sum m_{i,j}\Delta_i \leq \lambda$

If there is a solution with $\Delta_0 \geq \alpha$, we can give two distributions N_0 and N_1 as follows. Let R be the sum of the positive Δ_i. Note that $n_1 + 1 \geq \sum_i |\Delta_i| \geq R \geq \Delta_0 \geq \alpha$. N_0 is supported on those i with $\Delta_i > 0$, and the probability of strings with Hamming weight i solutions is Δ_i/R for such i. N_1 is supported on those i with $\Delta_i < 0$, and the probability of Hamming weight i is $-\Delta_i/R$.

For any i, the difference between the probabilities is Δ_i/R. Thus, the difference between their probabilities of the all-zero string is $\Delta_0/R \geq \alpha/(n_1 + 1)$, and the statistical distance between the noisy versions is: $\sum_j |\sum_i m_{i,j}\Delta_i/R| = 1/R\sum_j |\sum_i m_{i,j}\Delta_i| \leq \lambda(n_2 + 1)/R \leq \lambda(n_2 + 1)/\alpha$.

So if the optimum objective is greater than α, there are two distributions such that their noisy versions are indistinguishable to within $\lambda(n_2 + 1)/\alpha$. Thus, any $\alpha/(3(n_1 + 1))$ estimation algorithm will require $\alpha/((n_2 + 1)\lambda)$ samples.

If the optimum objective is less than α, consider the dual system of inequalities. Say we multiply the first equation by $w > 0$, the lower bound in the i'th example of the second set of inequalities by $u_i > 0$, the upper bound by $v_i > 0$, the lower bound in the j'th example in the third set of inequalities by $y_j > 0$ and the upper bound by $z_j > 0$. Then we get the induced inequality

$\sum_i (w + v_i - u_i + \sum_j (z_j - y_j)m_{i,j})\Delta_i \leq \sum_i (u_i + v_i) + \lambda \sum_j (y_j + z_j)$.

So the dual is to minimize $\sum_i (u_i + v_i) + \lambda \sum_j (y_j + z_j)$ subject to $w + v_0 - u_0 + \sum_j (z_j - y_j)m_{0,j} = 1$ and $w + v_i - u_i + \sum_j (z_j - y_j)m_{i,j} = 0$ for $n_1 \geq i \geq 1$. Note that if the objective is less than α, then so is each u_i and v_i, each y_j and z_j are at most α/λ, and $w < |v_1 - u_1| + max_j |z_j - y_j| = O(\alpha/\lambda)$.

Let $\beta_j = (z_j - y_j) + w$, and consider the algorithm : Let J be the random variable given by the observed type of the noisy sample. Estimate the expected value of β_J to within $O(\alpha)$. The constraints say that, on a type 0 input, the expectation of β_J is $1+O(\alpha)$ and on any other Hamming weight, the expectation is $O(\alpha)$ in absolute value.

Thus, the expectation of β_J will be within $O(\alpha)$ of the probability of the original sample being type 0. Since each $|\beta_j|$ is at most $O(\alpha/\lambda)$, we can estimate the expectation to within α using $O(1/\lambda^2)$ samples. In the language of [3]. the vector $\boldsymbol{\beta}$ is a *local inverse* of M at 0. We summarize the discussion in the following:

Theorem 1. *Let M be any $n_1 \times n_2$ stochastic matrix, and let $0 < \alpha, \lambda < 1$. Then either there is an pair of (α, λ)-fooling distributions for M (and hence, any $\alpha/3$-estimator for the probability of 0 requires $\Omega(1/\lambda)$ samples) or there is*

a local inverse for M with maximum coefficients of size $O(n_1 n_2 \alpha/\lambda)$ (and hence there is a time polynomial in n_1, n_2 and $1/\lambda$ algorithm to estimate the probability of 0 within $O(\alpha)$).

Acknowledgments. We are grateful to Avi Wigderson for introducing us to the problem and for useful discussions and suggestions, and to Amir Yehudayoff for helpful comments on an earlier version. We thank Chris Beck, Shachar Lovett, Mike Saks, and Ankur Moitra for helpful conversations.

References

1. Belkin, M., Sinha, K.: Polynomial learning of distribution families. In: FOCS 2010, pp. 103–112 (2010)
2. Dasgupta, S.: Learning mixtures of gaussians. In: FOCS 1999, p. 634. Computer Society (1999)
3. Dvir, Z., Rao, A., Wigderson, A., Yehudayoff, A.: Restriction access. In: Innovations in Computer Science 2012, pp. 19–33 (2012)
4. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: STOC 1989, pp. 25–32 (1989)
5. Karp, R.M., Shenker, S., Papadimitriou, C.H.: A simple algorithm for finding frequent elements in streams and bags. ACM Trans. Database Syst. 28(1), 51–55 (2003)
6. Kearns, M., Mansour, Y., Ron, D., Rubinfeld, R., Schapire, R.E., Sellie, L.: On the learnability of discrete distributions. In: STOC 1994, pp. 273–282 (1994)
7. Moitra, A., Saks, M.: A polynomial time algorithm for lossy population recovery. Manuscript (2013)
8. Moitra, A., Valiant, G.: Settling the polynomial learnability of mixtures of gaussians. In: FOCS 2010, pp. 93–102 (2010)
9. Arora, S., Kannan, R.: Learning mixtures of arbitrary gaussians. In: STOC 2001, pp. 247–257 (2001)
10. Wigderson, A., Yehudayoff, A.: Population recovery and partial identification. In: FOCS 2012, pp. 390–399 (2012)
11. Woeginger, G.J.: When does a dynamic programming formulation guarantee the existence of an fptas? In: SODA 1999, pp. 820–829 (1999)

Prasad Raghavendra Sofya Raskhodnikova
Klaus Jansen José D.P. Rolim (Eds.)

Approximation, Randomization, and Combinatorial Optimization

Algorithms and Techniques

LNCS 8096

16th International Workshop, APPROX 2013
and 17th International Workshop, RANDOM 2013
Berkeley, CA, USA, August 2013, Proceedings

Springer

Lecture Notes in Computer Science 8096

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Prasad Raghavendra Sofya Raskhodnikova
Klaus Jansen José D.P. Rolim (Eds.)

Approximation, Randomization, and Combinatorial Optimization

Algorithms and Techniques

16th International Workshop, APPROX 2013
and 17th International Workshop, RANDOM 2013
Berkeley, CA, USA, August 21-23, 2013
Proceedings

Springer

Volume Editors

Prasad Raghavendra
University of California
Berkeley, CA, USA
E-mail: prasad@cs.berkeley.edu

Sofya Raskhodnikova
Pennsylvania State University
University Park, PA, USA
E-mail: sofya@cse.psu.edu

Klaus Jansen
University of Kiel
Kiel, Germany
E-mail: kj@informatik.uni-kiel.de

José D.P. Rolim
University of Geneva
Carouge, Switzerland
E-mail: jose.rolim@unige.ch

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-40327-9 e-ISBN 978-3-642-40328-6
DOI 10.1007/978-3-642-40328-6
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013944978

CR Subject Classification (1998): F.2.2, G.2.2, G.2.1, F.1.2, G.1.0, G.1.2, G.1.6, G.3, E.1

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at the 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2013) and the 17th International Workshop on Randomization and Computation (RANDOM 2013), which took place concurrently in UC Berkeley, during August 21–23, 2013. APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally difficult problems, and was the 16th in the series after Aalborg (1998), Berkeley (1999), Saarbrücken (2000), Berkeley (2001), Rome (2002), Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), Princeton (2011), and Boston (2012). RANDOM is concerned with applications of randomness to computational and combinatorial problems, and was the 17th workshop in the series following Bologna (1997), Barcelona (1998), Berkeley (1999), Geneva (2000), Berkeley (2001), Harvard (2002), Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), Princeton (2011), and Boston (2012).

Topics of interest for APPROX and RANDOM are: design and analysis of approximation algorithms, hardness of approximation, small space algorithms, sub-linear time algorithms, streaming algorithms, embeddings and metric space methods, mathematical programming methods, combinatorial problems in graphs and networks, game theory, markets and economic applications, geometric problems, packing, covering, scheduling, approximate learning, design and analysis of online algorithms, design and analysis of randomized algorithms, randomized complexity theory, pseudorandomness and derandomization, random combinatorial structures, random walks/Markov chains, expander graphs and randomness extractors, probabilistic proof systems, random projections and embeddings, error-correcting codes, average-case analysis, property testing, computational learning theory, and other applications of approximation and randomness.

The volume contains 23 papers, selected by the APPROX Program Committee out of 46 submissions, and 25 papers, selected by the RANDOM Program Committee out of 52 submissions. In addition to presentations on these papers, the program included invited talks by Persi Diaconis (Stanford University, USA), Luca Trevisan (Stanford University, USA), and Santosh Vempala (Georgia Tech, USA).

We would like to thank all of the authors who submitted papers, the invited speakers, the members of the Program Committees, and the external reviewers. We gratefully acknowledge the support from the Computer Science Division,

University of California, Berkeley, the Department of Computer Science and Engineering at the Pennsylvania State University, the Institute of Computer Science of the Christian-Albrechts-Universität zu Kiel and the Department of Computer Science of the University of Geneva.

August 2013

Prasad Raghavendra
Sofya Raskhodnikova
Klaus Jansen
José D.P. Rolim

Organization

Program Committees

APPROX 2013

Nikhil Bansal	Eindhoven University, The Netherlands
Chandra Chekuri	University of Illinois, Urbana-Champaign, USA
Eden Chlamtáč	Ben-Gurion University, Israel
Nikhil Devanur	Microsoft Research Redmond, USA
Uriel Fiege	Weizmann Institute, Israel
Claire Matheiu	Brown University, USA
Ankur Moitra	Institute of Advanced Study, Princeton, USA
Seffi Naor	Technion, Israel
Yuval Rabani	Hebrew University, Israel
Prasad Raghavendra	University of California, Berkeley, USA (Chair)
Roy Schwartz	Microsoft Research Redmond, USA
Mohit Singh	Microsoft Research Redmond, USA
Ola Svennson	École Polytechnique Fédéral de Lausanne, Switzerland
Mohammed Taghi Hajighayi	University of Maryland, College Park, USA
Madhur Tulsiani	Toyota Technical Institute Chicago, USA
Rico Zenklusen	John Hopkins University, USA

RANDOM 2013

Amit Chakrabarti	Dartmouth College, USA
Nikolaos Fountalakis	University of Birmingham, UK
Ariel Gabizon	Technion, Israel
Parikshit Gopalan	Microsoft Research, Silicon Valley, USA
Dan Gutfreund	IBM Research, Haifa, Israel
Prahladh Harsha	Tata Institute of Fundamental Research, India
Tomas Hayes	University of New Mexico, USA
Michael Krivelevich	Tel Aviv University, Israel
Shachar Lovett	University of California at San Diego, USA
Russell Martin	University of Liverpool, UK
Dieter van Melkebeek	University of Wisconsin-Madison, USA
Sofya Raskhodnikova	Pennsylvania State University, USA (Chair)
Shubhangi Saraf	Rutgers University, USA
Christian Sohler	TU Dortmund University, Germany
David P. Woodruff	IBM Research, Almaden, USA
Amir Yehudayoff	Technion, Israel

External Reviewers

Melika Abolhasani
Matthew Anderson
Antonios Antoniadis
Per Austrin
Yossi Azar
Mohammadhossein Bateni
Tugkan Batu
Arnab Bhattacharyya
Abhishek Bhowmick
Eric Blais
Michel Bode
Beate Bollig
Magnus Bordewich
Joshua Brody
Shahar Chen
Amin Coja-Oghlan
Graham Cormode
Nicholas Crawford
Varsha Dani
Anindya De
Martin Dietzfelbinger
Michael Dinitz
Anne Driemel
Andrew Drucker
Zeev Dvir
Ebrahim Ehsanfar
David Eppstein
Moran Feldman
Michael Forbes
Tom Friedetzky
Alan Frieze
David Galvin
David Gamarnik
Sumit Ganguly
Ran Gelles
Shayan Oveis Gharan
Elena Grigorescu
Sudipto Guha
Ankit Gupta
Patrick Hayden
Frank Hellweg
Piotr Indyk
Rahul Jain
Tali Kaufman
Shiva Kasiviswanathan
Neeraj Kayal
Gillat Kol
Guy Kortsarz
Dariusz Kowalski
Lap Chi Lau
Massimo Lauria
James Lee
Jon Lee
Virginie Lerays
Ke Li
Kostya Makarychev
David Malec
Yishay Mansour
Kevin Matulef
Andrew McGregor
Raghu Meka
Morteza Monemizadeh
Cris Moore
Benjamin Moseley
Wolfgang Mulzer
Alexander Munteanu
Viswanath Nagarajan
Ilan Newman Krzysztof Onak
Rasmus Pagh
Debmalya Panigrahi
Will Perkins
Preyas Popat
Eric Price
Ilya Razenshteyn
Ricardo Restrepo Lopez
Noga Ron-Zewi
Aaron Roth
Guy Rothblum
Atri Rudra
Sushant Sachdeva
Rahul Santhanam
Swagato Sanyal
Ramprasad Saptharishi
Melanie Schmidt
Chris Schwiegelshohn
Pranab Sen

Ronen Shaltiel
Makrand Sinha
Gregory Sorkin
Perla Sousi
Piyush Srivastava
Alexandre Stauffer
Chaitanya Swamy
Ning Tan
Amnon Ta-Shma
Kunal Talwar
Prasad Tetali
Mikkel Thorup
Greg Valiant
Paul Valiant
Danny Vilenchik
Emanuele Viola
Omri Weinstein
Benjamin Weitz
Udi Wieder
Ryan Williams
Sergey Yekhanin
Yuichi Yoshida
Qin Zhang

Table of Contents

APPROX

RANDOM