Information and Computation **167**, 27–45 (2001)
doi:10.1006/inco.2000.3012, available online at http://www.idealibrary.com on IDEAL®

# On the Decisional Complexity of Problems Over the Reals

Moni Naor* and Sitvanit Ruah

*Weizmann Institute of Science, Department of Applied Math and Computer Science, Rehovot 76100, Israel*
E-mail: naor@wisdom.weizmann.ac.il, sitvanit@wisdom.weizmann.ac.il

We consider the role of randomness for the decisional complexity in algebraic decision (or computation) trees, i.e., the number of comparisons ignoring all other computation. Recently Ting and Yao showed that the problem of finding the maximum of $n$ elements has decisional complexity $O(\log^2 n)$ (1994, *Inform. Process. Lett.*, **49**, 39–43). In contrast, Rabin showed in 1972 an $\Omega(n)$ bound for the deterministic case (1972, *J. Comput. System Sci.*, **6**, 639–650). We point out that their technique is applicable to several problems for which corresponding $\Omega(n)$ lower bounds hold. We show that in general the randomized decisional complexity is logarithmic in the size of the decision tree. We then turn to the question of the number of random bits needed to obtain the Ting and Yao result. We provide a deterministic $O(k \log n)$ algorithm for finding the elements which are larger than a given element, given a bound $k$ on the number of these elements. We use this algorithm to obtain an $O(\log^2 n)$ random bits and $O(\log^2 n)$ queries algorithm for finding the maximum. 

## 1 INTRODUCTION

The power of probabilistic models of computation has been studied extensively since the introduction of randomization to algorithms. The main reason for adding randomization is to obtain more efficient algorithms. In addition to time and space, a natural measure for the complexity of an algorithm is the *decisional complexity* which corresponds to the number of conditional statements (also called queries) performed for the worst case input. As it turns out, when considering decision and computation trees significant gaps exist between randomized and non-randomized decisional complexity classes: In the algebraic decision tree and algebraic computation tree models it is known [Rab72, Jar81, MPR94] that the deterministic decisional complexity of finding the maximum of $n$ real numbers is $\Omega(n)$. Conversely, the co-nondeterministic decisional complexity of the problem (the smallest number of queries required to prove that an element is not maximal) in this model is $O(1)$. Ting and Yao [Ting93, TY94] used the small co-nondeterministic complexity of the problem to construct a randomized algorithm that solves the problem using $O(\log^2 n)$ queries and $O(n \log^2 n)$ random bits with $O(1/n^c)$ error.

These gaps make the decisional complexity one of the few measures where randomness is provably exponentially powerful. Such gaps cannot be obtained in all versions of the ACT and ADT models. In models that use only bounded degree queries, or examine only a bounded number of elements in a query, there are results showing that the decisional complexity of many computation problems does not reduce much by adding randomness and even a small probability of error (see [MT85, S85, Ni91, Gri99]). In this paper we focus our attention on the power of randomness in decisional complexity: for what problems are the results of Ting and Yao [Ting93, TY94] applicable; how much randomness is really needed.

Our main results are in showing that randomness can be limited to $O(\log^2 n)$ random bits.

We present a probabilistic algorithm for finding the maximum of $n$ distinct elements with $O(1/n^c)$ error probability, using $O(\log^2 n)$ queries. The advantage of the algorithm is that it uses only $O(\log^2 n)$ random bits, thus improving the $O(n \log^2 n)$ randomness complexity of [TY94]. The algorithm makes use of a deterministic algorithm for finding the elements which are larger than a given element, given a bound $k$ on the number of these elements. Ting [Ting93], gave a probabilistic proof to the existence of an $O(k^2 \log n)$ deterministic algorithm for this later problem. We observe that only $O(k \log n)$ queries are actually needed and then turn Ting's non-constructive scheme into a completely explicit algorithm with $O(k \log n)$ decisional complexity. This is done by applying small probability spaces.

* Research supported by a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences.



0890-5401/01 $35.00

The ideas of [TY94] for the *maximal element* problem are applied to derive probabilistic algorithms with $O(\log n)$ decisional complexity and $O(1/n^c)$ error probability also for other problems. On the other hand, we describe how to obtain $\Omega(n)$ deterministic, nondeterministic, and randomized with no error lower bounds for these problems using the results of [Rab72, Jar81, MPR94]. The motivation is to show more examples for the gaps between the deterministic and probabilistic decisional complexities. In addition, we show how to reduce the randomness complexity of these problems from $O(n \log n)$ to $O(\log n)$. We also relate the *size decisional complexity* (the number of leaves in the smallest tree) and the randomized decisional complexity and show that the latter is logarithmic in the former.

Note that there are applications in which the decisional complexity has a special role, for example, in the technique of *prefetching* and in *automated parallelizing*. In prefetching a block of data is brought into memory before it is actually referenced. In straight-line algorithms, in which only computations are performed, prefetching is possible because we know what the next statement is, before the current statement is executed. On the other hand, the step following a conditional statement depends on whether the condition holds or not, and hence cannot be fetched before the condition is tested. Straight-line algorithms are also generally easier to parallelize than algorithms that involve queries. Note however that not every algorithm whose decisional complexity is small is suitable for these applications, and the price of computations should be considered as well.

### 11. Organization

The paper is organized as follows: Section 2 defines the computation models we use. Section 3 summarizes related work. Section 4 discusses the gaps between the deterministic and probabilistic decisional complexities introduced in [TY94, Rab72, Jar81, MPR94]. The ideas presented in these papers are applied to exhibit gaps between the deterministic and probabilistic decisional complexities also for each of the problems: *simultaneous positivity, direct oriented convex hull, successive elements*, and *sorted list* (the exact definitions are given in Section 4.1). In addition, we show how to reduce the randomness complexity of these problems from $O(n \log n)$ to $O(\log n)$. We also relate the size decisional complexity (the number of leaves in the smallest tree) and the randomized decisional complexity.

Section 5 describes how to turn Ting's non-constructive scheme for finding the $k$-largest elements into a completely explicit algorithm with $O(k \log n)$ decisional complexity.

Section 6 describes a probabilistic algorithm for finding the maximum of $n$ distinct elements with $O(1/n^c)$ error probability, using $O(\log^2 n)$ queries.

## 2 MODEL OF COMPUTATION AND RELATED NOTATIONS

Given an input $x$ we consider the problems of deciding if $x$ belongs to a given set $W$ (decision problem) or finding some elements of the input that satisfy a given property (search problem).

We find it easier to describe algorithms explicitly in a model that allows computation steps. Hence, our model of computation is an *algebraic computation tree* (ACT), a rooted binary tree with three kinds of nodes: computation nodes, query nodes, and leaves. In *computation nodes*, a computation $z_v \leftarrow f(z_1 \ldots, z_m)$ is executed, where $f$ is a rational function ($f$ can be written as $f(z_1, \ldots, z_m) = p(z_1, \ldots, z_m)/q(z_1, \ldots, z_m)$, where $p$ and $q$ are polynomials), and the $z_i$'s are either the input elements or variables computed in some lower level. These nodes have just one emanating edge. In a *query node*, a query $z_v \mu 0$ is performed where $\mu \in \{=, \geq, >\}$ and $z_v \in \{x_1, \ldots, x_n\}$ or $z_v$ was computed in a lower level node. The node has two emanating edges, for the two possible outcomes of the query. A leaf is labeled with an output value. For a decision problem, this value is just "1" or "0." For a search problem, this value is the element or subset of elements that satisfy the required property.

A nondeterministic algebraic computation tree (NACT) is a rooted binary tree that has the three types of nodes of an ACT and also *guessing nodes* in which a nondeterministic choice is made. These nodes have two unlabeled emanating edges. For every input $x \in R^n$, and every leaf in the NACT that $x$ can reach, the label of the leaf is the correct output value for $x$.

A *probabilistic algebraic computation tree* (PACT) is a rooted binary tree that has the three types of nodes of an ACT and also *probabilistic nodes*, in which a random bit $\in \{0, 1\}$ is chosen with equal probability. A probabilistic node has two emanating edges, for the two possible bits. At each execution of the algorithm, the sequence of random bits chosen at the probabilistic steps forms a random string

$r$. $T$ is a distribution over deterministic ACTs. For each possible random string $r$, a deterministic $T_r$ is executed. A PACT $T$ solves a problem with error probability $\alpha$ if $\forall x \in R^n$

$$\Pr[T \text{ gives the correct output value on input } x \mid \geq 1 - \alpha.$$

A set $W$ is accepted by a PACT $T$ with *one-sided error* $\alpha$ if for $x \in W$

$$\Pr[T \text{ accepts } x] \geq 1 - \alpha$$

and for $x \notin W$

$$\Pr[T \text{ accepts } x] = 0.$$

A set $W$ is accepted by PACT $T$ with *two-sided error* $\alpha$ if for $x \in W$

$$\Pr[T \text{ accepts } x] \geq 1 - \alpha$$

and for $x \notin W$

$$\Pr[T \text{ accepts } x] \leq \alpha.$$

We next recall from [MPR94, Definition 2.1] the formal definitions of the complexity measures to be used: Let $T$ be an algebraic computation tree and $W \subseteq R^n$.

- The *decisional height* of a path $P$ in $T$, $h_D(P)$, is the number of query nodes on $P$.
- The *decisional height of* $T$ is the maximum over all paths $P$ in $T$ of $h_D(P)$.
- The *decisional complexity of* $W$, $C_D(W)$, is the minimum decisional height of all ACTs that decide on membership in $W$.

Analogous measures are defined for PACTs and NACTs: let $T$ be a probabilistic algebraic computation tree, $W \subseteq R^n$ and $x \in R^n$. Let $\hat{T}_r$ be the ACT executed for random string $r$. Denote by $\hat{T}_r(x)$ the path that $x$ follows in $\hat{T}_r$. The *randomized decisional height* of $T$ is:

$$RC_D(T) = \max_{x \in R^n} \mathop{\mathrm{E}}_{r}(h_D(\hat{T}_r(x))).$$

The *randomized with no error decisional complexity* of $W$, $RC_D(W)$, is the minimum decisional complexity over all PACTs that decide correctly on membership in $W$. $RC_D(W)$ corresponds to the complexity of Las Vegas algorithms.

Another measure considered for PACTs is the *randomness complexity* of $T$. This is the maximum over all paths $P_\ell$ of $T$ of the number of probabilistic nodes on $P_\ell$, or equivalently, the length of the longest random string.

For an NACT $T$, again let $\hat{T}_r$ be the ACT executed for choice string $r$ (where $r$ is the concatenation of the bits chosen in the guessing nodes of $T$) and let $\hat{T}_r(x)$ be the path that $x$ follows in $\hat{T}_r$. The *nondeterministic decisional height* of $T$ is $NT_D(T) = \max_{x \in R^n} \min_r(h_D(\hat{T}_r(x)))$. The *nondeterministic decisional complexity* of $W \subseteq R^n$, $NT_D(W)$, is the minimum height of all NACTs that decide on membership in $W$. The corresponding complexity measures for search problems are defined similarly.

Another interesting measure of complexity is the decisional size of an ACT $T$ which is the number of leaves in $T$. We use this measure in connection with the model of ternary algebraic computation trees, i.e., where at each node there is a three-way split according to $<, >$, or $=$. Each *binary* ACT has an equivalent *ternary* ACT, constructed in the obvious way. The decisional size complexity of $W \subseteq R^n$, $C_S(W)$, is the minimum decisional size over all ternary ACTs that decide membership in $W$. We relate the randomized decisional complexity and size complexities of ternary ACTs, by using the fact that in a ternary ACT there is a way to test (with high probability) if a given input reaches a certain leaf. However, this technique does not apply anymore when we consider *binary* ACTs, because of the

difficulty of distinguishing inputs which are roots of some of the polynomials on the path from the root to the leaf, but do not reach that leaf.

Another model of computation commonly used in the literature is the *algebraic decision tree* (ADT), which includes only query nodes and leaves (i.e., has no computation nodes). Its nondeterministic and probabilistic versions are the NADT and PADT, respectively, defined similarly to NACT and PACT.

## 3 RELATED WORK

The two main models considered in the study of computational complexity over the reals are the algebraic computation tree and the algebraic decision tree. The set of functions that can be computed at a computation node in the ACT or tested at a query node in the ADT varies from one version of these models to another.

Some geometric techniques were developed for obtaining lower bounds for decision problems in these models. Examples of these are the region counting argument of Dobkin and Lipton [DL79] and the flat counting of Rivest and Yao [RY80] for linear decision trees and the connected components counting of [SY82, BenOr83] for ADTs and ACTs with arithmetic functions.

Rabin [Rab72] studied the decisional complexity of membership problems represented by a conjunction of linear forms. He proved a linear lower bound on the decisional complexity of a restricted type of ADT for these problems. Jaromczyk [Jar81] showed these lower bounds still hold in the more general case of polynomial forms. The generalization of these lower bounds to the broader class of ACTs is proved in [MPR94]. The next section describes these results and some extensions of them.

Similar results were proved also for randomized and nondeterministic algorithms. Meyer Auf Der Heide [Mey85a, Mey85b] showed that the deterministic and probabilistic complexities of a problem are polynomially related. He proved that a PACT which accepts $L \subseteq R^n$, in expected time $t$, can be simulated by a deterministic ACT in $O(t^2 n)$ steps.

Manber and Tompa [MT85] gave examples for problems with $\Omega(n \log n)$ deterministic and nondeterministic decisional complexities but $O(\mathit{polylog})$ co-non-deterministic decisional complexity in the linear ADT model. One of their results in probabilistic models is an $\Omega(n)$ lower bound for deciding maximality of an element by an ADT that examines a bounded number of elements in each query.

Snir [S85] generalized the arguments of [DL79] to one-sided error linear PADTs. He gave a linear lower bound on deciding maximality of an element in this model. The component counting argument is generalized for lower bounds on two-sided error linear PADT in [Mey85b].

Grigoriev [Gri99] proved an $\Omega(\log N)$ lower bound for probabilistic computation trees recognizing an arrangement (i.e., a union of hyperplanes) with $N$ faces. This is applied to give an $\Omega(n^2)$ lower bound for the *Knapsack* problem and an $\Omega(n \log n)$ lower bound for the *element distinctness* problem in the PACT model.

The decisional size complexity is considered by Grigoriev *et al.* [GKY95], who obtain an exponential lower bound for the maximal element problem in bounded degree ADTs.

In the Boolean decision tree model, Nisan [Ni91] shows that even allowing error does not help much in reducing the randomized decisional complexity of problems with small non-deterministic *and* co-non-deterministic decisional complexities.

However, there are some examples for the usefulness of randomization in algorithms. Snir [S85] introduced a family of problems $P_n$ that take $O(3^n)$ time in the probabilistic linear ADT but cannot be solved by less than $O(4^n)$ in the deterministic linear ADT.

Buergisser *et al.* [BKL93] describe an $O(n)$ probabilistic algorithm for testing membership in the set $\{(x, y) \in R^{2n} \mid y \text{ is a permutation of } x\}$. They use the ACT model where each arithmetic operation is counted. The deterministic complexity of this problem is $\Omega(n \log n)$.

Ting and Yao [Ting93, TY94] improved the upper bound on the randomized decisional complexity of finding the maximum of $n$ distinct elements. They gave an $O(\log n)$ Monte Carlo algorithm for deciding maximality, and an $O(\log^2 n)$ Monte Carlo algorithm for finding the maximum.

Ben-Or [BenOr96a] proved the optimality of Ting and Yao's algorithm for deciding maximality. He showed that for $k < n - 1$ any randomized algorithm for verifying that $x_1 = \max\{x_1, \ldots, x_n\}$ using at most $k$ comparisons of analytic functions must have error probability greater than $1/2^k$. In addition, [BenOr96a] shows that any randomized algorithm with small error probability for verifying that $x_1$ is the *median* of $x_1, \ldots, x_n$ requires $\Omega(n)$ comparisons of analytic functions.

Wigderson and Yao [WY96] considered the number of *subset minimum tests* required for *finding* the maximum. A subset minimum test is of the form "$x < V$" (namely is $x \in X$ smaller than all elements in $V \subseteq X$?). They proved that $\Omega(\log^2 n)$ such tests are required for finding the maximum of $n$ elements. Ben-Or [BenOr96b] has also showed how to find the minimum explicitly, for the case that the input elements are not necessarily distinct. A description of this algorithm is given in Section 4.3.1.

## 4 EXTENSIONS TO KNOWN RESULTS

### 41. Lower Bounds for Error-less Algorithms

The sets accepted by an ACT coincide with the class of semi-algebraic sets. A set $W \subseteq R^n$ is semi-algebraic if it can be described as a boolean combination of polynomial equalities and inequalities, i.e., the set $W$ can be given as $W = \bigcup_{i \in I} \{x \in R^n \mid p_i(x) = 0, q_{i,j}(x) > 0 \text{ for } j \in J\}$, where $I$, $J$ are finite sets of positive integers (possibly empty), and $p_i, q_{i,j} \in R[X_1, \ldots, X_n]$.

By [BCR87, Theorem 2.7.1], for every closed semi-algebraic subset $W \subset R^n$ there are positive integers $k, t$ and polynomials $p_{i,j} \in R[X_1, \ldots, X_n]$ s.t.

$$W = \bigcup_{i=1}^{t} \{x \in R^n \mid p_{i,1}(x) \geq 0, \ldots, p_{i,k}(x) \geq 0\}. \tag{1}$$

The *width* of $W$ in $R^n$, $w(W, R^n)$, is the minimum non-negative integer $k \in N$ for which such a representation exists.

Rabin [Rab 72] defined the notion of a *complete proof* for $x \in W$, where $W$ is a closed semi-algebraic set of the form $W = \{x \in R^n \mid \ell_1(x) \geq 0, \ldots, \ell_m(x) \geq 0\}$ and $\ell_j$ is linear $1 \leq j \leq m$. A complete proof for $\ell_1(x) \geq 0, \ldots, \ell_m(x) \geq 0$ is a matrical representation of Eq. (1). Using these notations, Rabin proved a lower bound on the width of a complete proof for $\ell_1(x) \geq 0, \ldots, \ell_m(x) \geq 0$, which is equal to the width of $W$ in $R^n$.

Montana *et al.* [MPR94] applied the equivalence relation "generically equal" in order to prove lower bounds for general ACTs that accept a semi-algebraic set. They showed that in order to bound the decisional complexity of a semi-algebraic $W$, it is enough to give a lower bound on the width of closed semi-algebraic sets which are generically equal to $W$.

Two semi-algebraic $W, W' \subseteq R^n$ are generically equal if there exists a polynomial $q \in R[X_1, \ldots, X_n]$ s.t. the two sets are equal, except maybe for points which are roots of $q$. That is,

$$W \backslash \{x \in R^n \mid q(x) = 0\} = W' \backslash \{x \in R^n \mid q(x) = 0\}.$$

Montana *et al.* [MPR94] defined the notion *generic width* and showed it is a lower bound on the decisional complexity of any ACT that accepts $W \in R^n$. This notion applies to any semi-algebraic $W$ and not just closed. The generic width of a semi-algebraic $W$ in $R^n$, $\mathrm{w}_{\mathrm{gen}}(W, R^n)$, is

$$\mathrm{w}_{\mathrm{gen}}(W, R^n) = \min\{w(C, R^n) : C \text{ is closed, generically equal to } W \text{ in } R^n\}.$$

They established the connection between the generic width of a semi-algebraic set and the complexity of its membership problem in the following proposition:

PROPOSITION 4.1 [MPR94, Proposition 4.1]. *Let $W$ be a semi-algebraic subset of $R^n$. Then,*

$$\mathrm{w}_{\mathrm{gen}}(W, R^n) \leq C_D(W).$$

They bounded the generic width of semi-algebraic sets which are defined by a conjunction of polynomial inequalities, satisfying some condition as required in the following propositions:

Recall the *Jacobian matrix* $J$ of $p_1(x_1, \ldots, x_n), \ldots, p_m(x_1, \ldots, x_n)$ at a point $\alpha$ is the $m \times n$ matrix of which the $i$th row is the vector of the partial derivatives of $p_i(x_1, \ldots, x_n)$ evaluated at $\alpha$. That is, $J = (a_{i,j})_{m \times n}$, where $a_{i,j} = (\partial p_i / \partial x_j)(\alpha)$.

PROPOSITION 4.2 [MPR94, Corollary 3.9]. *Let $p_1(X), \ldots, p_m(X) \in R[X_1, \ldots, X_n]$ be a collection of polynomials, and let $\alpha$ be a point in $R^n$ such that $p_1(\alpha) = 0, \ldots, p_m(\alpha) = 0$ and the rank of the Jacobian matrix defined by $p_1(X), \ldots, p_m(X)$ at $\alpha$ is $m$, i.e., rank $J(p_1, \ldots, p_m)_\alpha = m$.*

*Let* $W = \{x \in R^n \mid p_1(x) \geq 0 \ldots, p_m(x) \geq 0\}$. *Then,* $w_{\text{gen}}(\{x \in R^n \mid p_1(x) \geq 0, \ldots, p_m(x) \geq 0\}, R^n) = m$.

PROPOSITION 4.3 [MPR94, Corollary 3.10]. *Let* $f_1, \ldots, f_m \in R[X_1, \ldots, X_n]$ *be a collection of polynomials for which there is a point* $\alpha \in R^n$ *and a positive integer* $1 \leq k < m$ *s.t.*

$$f_1(\alpha) = \cdots = f_k(\alpha) = 0,\ f_{k+1}(\alpha) > 0, \ldots, f_m(\alpha) > 0$$

*and the rank of the Jacobian matrix verifies* $J(f_1, \ldots, f_k)_\alpha = k$.

*Let* $W = \{x \in R^n \mid f_1(x) \geq 0, \ldots, f_m(x) \geq 0\}$. *Then,* $k \leq w_{\text{gen}}(W, R^n)$.

They applied these conditions to derive linear deterministic lower bounds on the decisional complexities of the problems *maximal element, simultaneous positivity,* and *direct oriented convex hull* defined in the sequel. This is also applicable for the problems *sorted list* and *successive elements* listed below.

We claim that the notions of *width* and *generic width* are not only lower bounds for deterministic algorithms, but also for non-deterministic algorithms and probabilistic algorithms that never err.

The complexity of a non-deterministic ACT is defined in terms of the minimal path length, and that of a probabilistic ACT in terms of the expectation of the lengths of the paths that an input follows. For that reason we first define the concept of *minimal width* which relates to the length of the *shortest* path the worst input follows. Later, we show that actually the *minimal width* of $W$ is equal to its *width*.

DEFINITION 4.1. Let $W$ be a semi-algebraic subset of $R^n$, $W$ closed. The minimal width of $W$ in $R^n$, $minw(W, R^n)$, is the minimum non-negative integer $s \in N$ s.t. there is $t$, and for $1 \leq i \leq t$ there are $k(i) \in N$ and polynomials $p_{i,j} \in R[X_1, \ldots, X_n]$ for $1 \leq j \leq k(i)$, s.t.

$$W = \bigcup_{i=1}^{t} \{x \in R^n \mid p_{i,1}(x) \geq 0, \ldots, p_{i,k(i)}(x) \geq 0\},$$

and $\max_{x \in W} \min_{i \in I(x)} k(i) = s$, where $I(x) = \{i \mid p_{i,1}(x) \geq 0, \ldots, p_{i,k(i)}(x) \geq 0\}$.

If $W$ is an open set, then the minimal width of $W$ is defined by replacing "$p_{i,j}(x) \geq 0$" By "$p_{i,j}(x) > 0$." Again, $minw(R^n, R^n) = minw(\emptyset, R^n) = 0$.

As defined in the Introduction, a non-deterministic ACT is said to accept $x$ within time $t$ if and only if there is an accepting path of length at most $t$ that $x$ follows. As will be seen, for the worst case input $x \in W$, this time is bounded from below by the generic width of $W$ in $R^n$. Hence this measure is a lower bound on the non-deterministic complexity of the membership problem for $W$, and consequently also for its randomized with no error complexity. The following claim shows that the concepts of *width* and *minimal width* are identical.

CLAIM 4.1. *For every semi-algebraic subset* $W \subseteq R^n$, $minw(W, R^n) = w(W, R^n)$.

*Proof.* Obviously $minw(W, R^n) \leq w(W, R^n)$. For the other direction, let $w(W, R^n) = k$. Assume $minw(W, R^n) < k$. Then, there is a representation

$$W = \bigcup_{i=1}^{t} \{x \in R^n \mid p_{i,1}(x) \geq 0, \ldots, p_{i,k(i)} \geq 0\},$$

and for every $x \in R^n$ there is $i$ s.t. $p_{i,1}(x) \geq 0 \wedge \cdots \wedge p_{i,k(i)}(x) \geq 0$, and $k(i) < k$. By possibly reordering the indices $i$, let $1 \leq i \leq s$ be the indices s.t. $k(i) < k$, but then

$$W = \bigcup_{i=1}^{s} \{x \in R^n \mid p_{i,1}(x) \geq 0, \ldots, p_{i,k(i)}(x) \geq 0\},$$

and $k(i) < k$ for every $i$ contradicting the fact that $w(W, R^n) = k$. ■

The following theorem gives a lower bound for the nondeterministic and randomized with no error decisional complexity of the membership problem of a semi-algebraic set $W$. Recall that the randomized

decisional height of a PACT $T$ is the expected value of the decisional height of the path that the worst input follows:

$$RC_D(T) = \max_{x \in R^n} \mathop{\mathrm{E}}_{r}(h_D(\hat{T}_\tau(x))).$$

The randomized with no error decisional complexity of $W$, $RC_D(W)$, is the minimum decisional height of all PACTs that decide on membership in $W$ with no error.

Obviously, $RC_D(W) \geq NT_D(W)$.

THEOREM 1. *Let $W \subseteq R^n$ be a semi-algebraic set; then*

$$NT_D(W) \geq \mathrm{w}_{\mathrm{gen}}(W, R^n).$$

*Proof.* Let $\mathrm{w}_{\mathrm{gen}}(W, R^n) = M$. If $M = 0$, the inequality trivially holds. Now, assume $M > 0$.

Let $\hat{T}$ be an NACT that accepts $W$. Let $P_1, \ldots, P_s$ be the accepting paths of $\hat{T}$. So for every $i$ the set of points $W(P_i)$ accepted at the leaf ending $P_i$ is a semi-algebraic set, and $W = W(\hat{T}) = \text{def} \bigcup_{j=1}^{s} W(P_j)$. Similarly to [MPR94] we produce a semi-algebraic set that is generically equal to $W$ by the following steps:

1. Eliminate every path that includes an equation test, i.e., a test of the form "$q(x) = 0$?" Let $P_{\ell_1}, \ldots, P_{\ell_t}$ be the remaining paths, then $W_1(\hat{T}) = \bigcup_{j=1}^{t} W(P_{i_j})$ is a semialgebraic set, and since we dropped only paths with equalities, $W_1(\hat{T})$ is generically equal to $W$ in $R^n$.

2. Replace every strict inequality $p(x) > 0$ on the paths defining $W_1(\hat{T})$ by a weak inequality $p(x) \geq 0$. Again the obtained set $W_2(\hat{T})$ is

$$W_2(\hat{T}) = \bigcup_{j=1}^{t} \{x \in R^n \mid p_{i,1}(x) \geq 0, \ldots, p_{i,k,(i)}(x) \geq 0\},$$

where $p_{i,1}, \ldots, p_{i,k,(i)}$ are the polynomials tested on path $P_{\ell_i}$. Hence, $W_2(\hat{T})$ is generically equal to $W_1(\hat{T})$ in $R^n$ and hence, also to $W$.

Denote by $h_D(\hat{T}_r(x))$ the decisional height of the path that $x$ follows in the NACT for random string $r$:

$$NT_D(\hat{T}) = \max_{x \in R^n} \min_{r} (h_D(\hat{T}_r(x))).$$

Since $M > 0$, there exists $x \in W$ that follows only paths that were not eliminated in steps 1 and 2; hence $NT_D(\hat{T}) \geq \mathit{minw}(W_2(\hat{T}), R^n) = w(W_2(\hat{T}), R^n) \geq M$. Since the above inequality is true for every NACT $\hat{T}$ that accepts $W$, we have that

$$NT_D(W) = \min_{\text{NACT } T \text{ for } W} NT_D(T) \geq M \qquad \blacksquare$$

A corollary from the above lower bound results is that even if all the inputs to the algorithm satisfy $Q(x) \neq 0$, for some polynomial $Q$, then at least $w_{\mathrm{gen}}(W, R^n) - 1$ queries are required to decide on membership in a semi-algebraic $W \subseteq R^n$ by an ACT, NACT, or PACT that never errs. In particular, this gives linear lower bounds on the number of queries required to solve problems as above, given that $p_i(x) \neq 0$ for every polynomial $p_i$ appearing in the conjunction of polynomials that defines $W$.

Each of the problems listed below are membership problems for subsets that are represented by a conjunction of inequalities of the form $p_j(x) \geq 0$ as above. Consequently, the deterministic, nondeterministic, and randomized with no error decisional complexities of each of them is $\Omega(n)$:

*Simultaneous positivity* (defined in [MPR94]): Given $n$ nonzero real numbers $x_1, \ldots, x_n$, decide whether $x_i \geq 0$ for $1 \leq i \leq n$.

*Direct oriented convex hull*: Given a sequence $(z_1, \ldots, z_n)$ of points in the real plane $z_i = (x_i, y_i)$, s.t. no three successive points (in cyclic order) lie on the same straight line, decide whether they are the clockwise oriented vertices of their convex hull. By [Jar81] this is the problem of testing membership

in the set

$$W = \{(z_1, \ldots, z_n) \in R^{2n} \mid d(z_1, z_2, z_3) \geq 0, \ldots, d(z_{n-2}, z_{n-1}, z_n) \geq 0,$$
$$d(z_{n-1}, z_n, z_1) \geq 0, d(z_n, z_1, z_2) \geq 0\},$$

where $d(z_i, z_k, z_j) = x_k(y_i - y_j) + y_k(x_j - x_i) + y_j \cdot x_i - y_i \cdot x_j$ and $\forall i\ d(z_i, z_{i+1}, z_{i+2}) \neq 0$.

*Maximal element*: Given a list of *distinct* real numbers $x_1, \ldots, x_n$, decide whether $x_1$ is the maximum.

*Sorted list*: Given a list of *distinct* real numbers $x_1, \ldots, x_n$, decide whether the list is sorted in increasing order.

*Successive elements*: Given $n$ *distinct* real numbers $x_1, \ldots, x_n$, decide whether $x_1$ and $x_2$ are successive in sorted order. This holds if and only if $(x_1 - x_k)(x_2 - x_k) \geq 0$ for $3 \leq k \leq n$.

## 42. Small-Bias Probability Spaces

In the sequel, small probability spaces are applied for constructing efficient deterministic algorithms and probabilistic algorithms with small decisional and randomness complexities. Specifically, we make use of the following types of random variables:

1. $\epsilon$-biased random variables [NN93] defined as follows: let $y_1, \ldots, y_n$ be 0, 1 random variables with joint probability distribution $D$. The variables $y_1, \ldots, y_n$ are said to be $\epsilon$-biased if for all subsets $U \subseteq \{1, \ldots, n\}$,

$$\left| \Pr_D \left[ \bigoplus_{j \in U} y_j = 0 \right] - \Pr_D \left[ \bigoplus_{j \in U} y_i = 1 \right] \right| \leq \epsilon.$$

For constant $\epsilon < 1/2$, the points of the probability space can be the columns of the generating matrix of an error correcting code that corrects a constant fraction of errors (for example Justesen codes [Jus72]). Sampling the resulting $\epsilon$-biased probability space requires $O(\log n + \log(1/\epsilon))$ random bits.

2. $k$-wise $\epsilon$-biased random variables defined as follows (Definition 2.2 in [NN93]): random variables $y_1, \ldots, y_n \in \{0, 1\}$ with joint probability distribution $D$ are $k$-wise $\epsilon$-biased if for every $U \subseteq \{1, \ldots, n\}$ such that $|U| \leq k$,

$$\left| \Pr_D \left[ \bigoplus_{j \in U} y_j = 0 \right] - \Pr_D \left[ \bigoplus_{j \in U} y_i = 1 \right] \right| \leq \epsilon.$$

In [NN93] there is a description of a construction of a $k$-wise $\epsilon$-biased probability space of size $(k \log n)/\epsilon^{O(1)}$. Hence, for constant $\epsilon$, this construction produces a probability space of size $O(k \log n)$.

3. $k$-wise $\delta$-dependent random variables: $\{0, 1\}$ random variables $y_1, \ldots, y_n$ with joint distribution $D$ are $k$-wise $\delta$-dependent if $\forall 1 \leq \ell \leq k, \forall S = \{i_1, \ldots, i_\ell\} \subseteq \{1, \ldots, n\}$,

$$\|D(S) - U(S)\| = \sum_{d \in \{0,1\}^\ell} \mid D((x_{i_1}, \ldots, x_{i_\ell}) = d) - U((x_{i_1}, \ldots, x_{i_\ell}) = d) \mid \leq \delta,$$

where $U$ stands for the uniform distribution.

In particular, for every $S \subseteq \{1, \ldots, n\}$ such that $|S| = i \leq k$, the probability that the random variables of $S$ attain a certain configuration deviates from $1/2^i$ by at most $\delta$. By [AGHP92, ABNNR92], such a probability space can be constructed, where the number of bits required to specify a point in the sample space is $O(\log \log n + k/2 + \log k + \log(1/\delta))$. Hence, for $\delta = O(1/n^c)$ and $k = O(\log n)$, sampling the resulting space requires $O(\log n)$ bits.

## 43. Monte Carlo Algorithms

### 43..1 *Decisional Complexity*

In contrast to the lower bounds of Section 4.1 for error-less algorithms, recent results of Ting and Yao [Ting93, TY94] present a Monte Carlo algorithm with $O(\log^2 n)$ decisional complexity for finding the maximum of $n$ distinct elements. They defined polynomial queries that can serve as a proof to the non-maximality of an element. They used the fact that for a uniformly random subset $S \subseteq \{1, \ldots, n\}$ (represented as a vector $(s_1, \ldots, s_n) \in \{0,1\}^n$), if $x_i$ is not the maximum, then it is equally likely that S contains an odd or even number of indices of elements larger than $x_i$. Denoting the set of indices of elements larger than $x_i$ by $G_i(x)$ (i.e., $G_i(x) = \{j \mid x_j > x_i\}$), that means,

$$\Pr\left[\prod_{j \in S \setminus \{i\}} (x_i - x_j) < 0\right] = \Pr\left[\bigoplus_{j \in S \cap G_i(x)} s_j = 1\right] = \frac{1}{2}.$$

On the other hand, if $x_i$ is the maximum, then a *parity test* $\prod_{j \in S\setminus\{i\}}(x_i - x_j) > 0$ on any subset $S$ gives a positive result. The error can be reduced to $1/n^c$ by choosing $O(\log n)$ subsets uniformly at random and accepting only if $x_i$ passed the parity tests on all these subsets.

This idea can be extended to other membership problems. Assume $W$ is represented as

$$W = \{x \in R^n \mid p_1(x) \geq 0, \ldots, p_m(x) \geq 0\}.$$

Assume we know $\prod_{j=1}^m p_j(x) \neq 0$, and we have a way to sample random subsets of polynomials $\{p_{j_1}, \cdots, p_{j_s}\}$. Then for $x \in W$, always $p_{j_1}(x) \cdots p_{j_s}(x) > 0$, and for $x \notin W$, $\Pr[p_{j_1}(x) \ldots p_{j_s}(x) < 0] = 1/2$.

Hence, we can execute the same procedure to decide if $x \in W$. That is, we sample uniformly at random $O(\log n)$ subsets of $\{1, \ldots, m\}$ and check if all the parity tests give positive results.

The algorithms of [Ting93, TY94] have polynomial randomness complexity ($O(n \log n)$ for checking maximality and $O(n \log^2 n)$ for finding the maximum). The amplification methods of random walks on expander graphs [AKS87, AS92, CW89, IZ89] and $\epsilon$-biased random variables [ABNNR92, AGHP92, NN93] enable us to reduce this randomness cost. We first recall the definitions required for the amplification methods we use:

A graph $G = (V, E)$ is called an $(n, d, c)$-expander if $|V| = n$, the maximum degree of a vertex is $d$, and for every set of vertices $W \subset V$ such that $(|W| \leq n)/2$, the inequality $|N(W)| \geq c|W|$ holds, where $N(W)$ denotes the set of all vertices in $V \setminus W$ adjacent to some vertex in $W$.

A $d$-regular expander is a $d$-regular graph $G(V, E)$ such that there exists $c$ for which $G$ is an $(n, d, c)$-expander.

Let $G = (V, E)$ be a $d$-regular expander where the absolute value of each of its eigenvalues (the eigenvalues of its adjacency matrix) but the first one is at most $\lambda$. Assume a one to one correspondence between $V$ and the set of all possible random strings of an $\epsilon$-biased probability space with random variables $y_1, \ldots, y_n$. By [LPS86, Mar88] such a graph can be constructed with degree $d$ and $\lambda \leq 2\sqrt{d-1}$ for each $d = p + 1$ where $p$ is a prime congruent to 1 modulo 4. The label of each node $v \in V$ is a characteristic vector of a subset $S_v \subseteq \{1, \ldots, n\}$. A "good" node for $i$ is a node which represents a subset with odd number of elements from $G_i(x)$. By the property of an $\epsilon$-biased probability space, we know that for every $i \in \{1, \ldots, n\}$ s.t. $i$ is not the maximum, there are at least $1/2 - \epsilon$ good nodes.

Instead of choosing $O(\log n)$ subsets uniformly at random, choose a random walk of length $O(\log n)$ on $G$ as described in [AS92, CW89, IZ89].

At each node on the random walk, test whether

$$\prod_{j \in S_v \setminus \{1\}} (x_1 - x_j) < 0,$$

where $v$ is the current node on the random walk. If one of the tests produced a positive answer, conclude $x_1$ is not the maximum. Otherwise, conclude it is. The following bound on the error probability is given in [AS92]:

THEOREM 2 [AS92, Corollary 2.8]. *Let $G = (V, E)$ be a $d$-regular graph on $n$ vertices, and suppose the absolute value of each of its eigenvalues but the first one is at most $\lambda$. Let $C$ be a set of $\delta n$ vertices of $G$. If*

$$((1-\delta)d^2 + \lambda^2)^{1/2} \leq \frac{d}{2^{1/4}},$$

*then, for every $\ell$, the probability that a randomly chosen walk of length $\ell$ avoids $C$ is at most $1/2^{\ell/4}$.*

By using the expander constructions of [LPS86, Mar88] and performing a random walk of length $4c \log n$, the above scheme produces an algorithm for deciding maximality with error probability $1/n^c$ (using Corollary 2.8 in [AS92]) and decisional and randomness complexities $O(\log n)$.

As before, the same method applies for deciding membership in any subset

$$W = \{x \in R^n \mid p_1(x) \geq 0, \ldots, p_m(x) \geq 0\}, \qquad \text{where } \prod_{j=1}^{m} p_j(x) \neq 0.$$

In particular the following theorem holds:

THEOREM 3. *For each of the problems simultaneous positivity, maximal element, direct oriented convex hull, successive elements, and sorted list (as defined in Section 4.1), and for every constant $c > 0$, an algorithm can be constructed that solves the problem with error probability $O(1/n^c)$, and has $O(\log n)$ decisional and randomness complexities.*

Ben-Or [BenOr96b] generalized the algorithm for finding maximum to the case where not all elements are distinct. He changed the procedure for checking if an element $x_i$ is maximal as follows:

- Find the number of elements $k$ equal to $x_i$.
- Choose $O(\log n)$ random subsets $S_1, \ldots, S_\ell \subseteq \{1, \ldots, n\}$.
- For each $1 \leq j \leq \ell$, find the number of elements in $S_j$ which are equal to $x_i$, then check if $S_j$ contains an odd number of elements larger than $x_i$. As before, if none of the subsets contained an odd number of elements, decide $x_i$ is the maximum.

Ben-Or used the fact that a subset $S$ contains at least $k$ elements equal to $y$ if and only if

$$\sum_{S \in A_{n-k+1}} \prod_{j \in S} (y - x_j)^2 = 0, \tag{2}$$

where $A_j = \{S \subseteq \{1, \ldots, n\} \| S| = j\}$. Hence, we can find the exact value of $k$ by a binary search. For checking if $S$ contains an odd number of elements, he tested the sign of the left-hand side of (2). By choosing random subsets of the right size, this procedure for checking maximality requires $O(\log k \log n)$ queries and $O(n \log n)$ random bits.

The results of [TY94, Rab72, MPR94] also imply the following example of a problem whose nondeterministic *and* co-non-deterministic decisional complexities are large, yet the randomized decisional complexity is small:

$P_1$: given distinct $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$, decide whether $x_1$ is the maximum of $x_1, \ldots, x_n$, *and* $y_1$ is *not* the maximum of $y_1, \ldots, y_n$.

PROPOSITION 4.4. *The nondeterministic and co-nondeterministic decisional complexities of the problem $P1$ are $\Omega(n)$, but the randomized (two-sided error) decisional complexity is $O(1)$.*

Obviously, there are problems that already their deterministic decisional complexity is low and much smaller than their *total complexity* (counting arithmetic computations). For example:

*Element distinctness:* Given $x_1, \ldots, x_n$ decide whether all elements are distinct.
*Set equality:* Given two sets $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$ decide whether the two sets are equal.

PROPOSITION 4.5. *The problems* element distinctness *and* set equality *have total complexity* $\Omega(n \log n)$, *yet their decisional complexity is only* $O(1)$.

*Proof.* Each of $x_1, \ldots, x_n$ is unique in the list if and only if $\prod_{i \neq j}(x_i - x_j) \neq 0$. For the case of *set equality*, $\{x_1, \ldots, x_n\} = \{y_1, \ldots, y_n\}$ if and only if $\prod_{\sigma \in S_n}(\sum_{j=1}^{n}(x_j - y_{\sigma(j)})^2) = 0$ (where $S_n$ is the permutation group of $n$ elements). ■

43..2 *Decisional Size Complexity*

As defined before, the decisional size complexity of an ACT $T$ is the number of leaves in $T$. We give a general relation between the randomized decisional and size complexities of decision problems (but not necessarily search problems) in the following theorem:

THEOREM 4. *A semi-algebraic set* $W \subset R^n$ *has two-sided error randomized decisional complexity which is* $O(\log C_S(W) + \log 1/\delta)$, *where* $C_S(W)$ *is the size decisional complexity of* $W$ *(and the size is measured for ternary trees) and* $\delta$ *is a bound on the error.*

*Proof.* We need the following proposition:

PROPOSITION 4.6. *Let* $T$ *be a ternary ACT that solves the membership problem for* $W$. *Given an input* $x$ *and a node* $v$ *in* $T$, *we can check with probability* $3/4$ *whether* $x$ *reaches* $v$.

*Proof.* Let $q_1, \ldots, q_s, p_1, \ldots, p_m$ be the polynomials s.t. $q_1(x) = 0, \ldots, q_s(x) = 0, p_1(x) > 0, \ldots, p_m(x) > 0$ on the path from the root to $v$. We can decide with probability $3/4$ and constant number of queries whether on input $x$ the path from the root to $v$ is traversed as follows:

$$\text{Choose random} S_1, S_2 \subseteq \{1, \ldots, m\} \text{ and check if } \prod_{j \in S_k} p_j(x) > 0, \text{ for } j = 1, 2 \text{ and } \sum_{i=1}^{s} q_i^2(x) = 0. \quad \blacksquare$$

Let $T$ be a ternary ACT that solves the membership problem for $W$, and let $x$ be an input. We utilize a separator decomposition technique used, e.g., in [NA91] for finding the leaf to which the computation on $x$ leads.

In a rooted tree $T$ on $n$ vertices, a node $v$ is called a *separator* if its removal from $T$ splits $T$ so that each connected component contains at most $\frac{2}{3}n$ of the nodes. By [Jo69, Meg83] each tree contains a separator.

Any tree $T$ has a *complete decomposition tree* $U$ on the same set of vertices [NA91]: $U$ is a rooted tree whose root $v$ is a separator of $T$; $v$'s children (in $U$) are the roots of the recursively defined separator decomposition trees for the connected components of $T$ resulting from the removal of $v$.

Let $U$ be a decomposition tree of $T$. In order to compute with unreliable tests, we extend $U$ as described in the *noisy comparison model* of Feige *et al.* [FRPU94]: each leaf is a parent of a chain of nodes of length $m' = O(\log(C_s(W)/\delta))$. Each node is labeled with the label of the leaf and has a pointer to the head of the chain.

Throughout the algorithm we use the following notations:

*u:* current node in $U$.

*t:* node in $T$ corresponding to $u$.

At any point we have to examine a given node and decide whether on input $x$ this node is reached. However, note that we may be in the wrong component altogether, so we should query on the parent (in the decomposition tree) of the current node as well.

- Let $m = c \log(C_s(W)/\delta)$, $m < m'$.
- $u \leftarrow$ root of $U$.
- Repeat for at most $m$ steps:

1. If $u$ is not the root of $U$, then check if we are in the right component, by repeating the test on the parent of $u$.

2. If we are in the wrong component, then go to the connected component corresponding to the"grandparent" (parent of the parent) of $u$.

3. Otherwise, decide if $u$ is an ancestor of $w$ by applying the technique described in Proposition 4.3.2 (if $u$ is a chain node, then choose tests of nodes above the chain).

4. If $u$ is an ancestor of $w$, then evaluate the test in $t$ to determine which of $u$'s children in $T$ is an ancestor of $w$, and assign it to $u$ (if $u$ is a chain node, then go the only child of $u$).

5. Otherwise, the connected component is the child of $u$ corresponding to the parent in $T$. $u$ is assigned this child (if $u$ is a chain node, then the connected component is the one corresponding to the grandparent of the head of the chain).

By Proposition 1 in [NA91], the height of the decomposition tree is $O(\log C_S(W))$. This is very similar to the noisy comparison trees model of Feige *et al.* [FRPU94] and the analysis there can be applied here to show that for probability of error $\delta$ you can find the correct leaf in time $O(\log C_S(W) + \log 1/\delta)$. The proof is as follows: Take a leaf $w$ in $U$, and suppose $x$ reaches a node in the connected component of $w$, but a leaf in $U$ corresponds to a connected component with one node, so $w$ is the leaf that $x$ reaches in $T$. Orient all the edges of $U$ towards $w$. So every node $v$ has exactly one outgoing edge, and all the other adjacent edges are directed towards $v$. The transition probability along the outgoing edge is at least $3/4$, and the transition probability of the incoming edges is at most $1/4$.

Let $m_f$ be a random variable counting the number of transitions in the direction of the edges and $m_b$ the number of backward transitions. So $m_b + m_f = m$. Since $m < m'$, the algorithm never reaches the end of a chain. We need to show that $m_f - m_b > \log C_s(W)$ with probability at least $1 - \delta$, implying that the correct chain is reached. We prove this by applying the following version of Chernoff bound [Chernoff52]:

THEOREM 5 (Chernoff bound). *Let $x_1, \ldots, x_n$ be independent $\{0, 1\}$ random variables with $Prob(x_i = 1) = p_i i = 1, \ldots, n$ and $\sum_{i=1}^{n} p_i > 0$. Let $X = \sum_{i=1}^{n} x_i$. Then for $0 < \epsilon < 1$*

$$Prob(X < (1 - \epsilon)E(X)) \leq e^{-E(X)\epsilon^2/2}.$$

In our case

$$x_i = \begin{cases} 1 & \text{a forward transition was performed at step } i \\ 0 & \text{otherwise} \end{cases}$$

$$Prob(x_i = 1) \geq \frac{3}{4}$$

$$m_f = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} Prob(m_f - m_b \leq \log C_s(W)) &= Prob\left(m_f \leq \frac{1}{2}(m + \log C_s(W)\right) \\ &= Prob\left(m_f \leq \frac{1}{2}(c \log(C_s(W)/\delta) + \log C_s(W)\right) \\ &\leq Prob\left(m_f \leq \left(\frac{1}{2}(c+1)\log(C_s(W)/\delta\right)\right) \\ &\leq Prob\left(m_f \leq \frac{2}{3}\frac{(c+1)}{c}E(m_f)\right) < \delta. \end{aligned}$$

For $c = (32 \log \frac{1}{\delta})/\log(C_s(W)/\delta)$. ■

## 5 FINDING THE $k$ LARGEST ELEMENTS

Assume we have a set of $n$ *distinct* elements $\{x_1, \ldots, x_n\}$ and an index $i$ s.t. at most $k$ elements from the set are larger than $x_i$. In this section we study the decisional complexity of finding these elements.

Since there are $\binom{n-1}{k}$ possible solutions to the problem, a trivial lower bound on this complexity is $\log(\binom{n-1}{k}) = \Omega(k \log(n/k))$.

Denote the set of possible inputs by $\tilde{R}^n = \{x = (x_1, \ldots, x_n) \in R^n \mid x_i \neq x_j \text{ if } i \neq j\}$. We say that $x_i$ is of rank $k+1$ if there are $k$ elements larger than $x_i$, i.e., $rank(x_i) = \mid G_i(x) \mid +1$, where $G_i(x) = \{j \mid x_j > x_i\}$. Let $x_i$ be s.t. $rank(x_i) > 1$. A good subset for $x_i$ is a subset of $\{1, \ldots, n\}$ that contains an odd number of elements from $G_i(x)$. Given such a subset $S$, the non-maximality of $x_i$ will be discovered by the query "$\prod_{j \in S \setminus \{i\}}(x_i - x_j) < 0$?"

Ting [Ting93] proved the existence of a collection of $O(k \log n)$ subsets s.t. for each $x$ with $1 < rank(x_i) \leq k+1$, there is a good subset in the collection. Given such a collection, Ting suggests an algorithm that is executed in phases as follows: in each phase test if $\prod_{j \in S \setminus \{i\}}(x_i - x_j) < 0$ for each $S$ in the collection. By the property of the collection, as long as not all elements of $G_i(x)$ are found, a good subset will be reached. Now, find an element of $G_i(x)$ using a binary search on the good subset; i.e., divide the subset into two and use a parity test to find which of the two halves contains an odd number of members of $G_i(x)$, and repeat the process until reaching a subset of one element $x_t$. Eliminate the element $x_t$ that was found (replace it with a very small element, say $-(\sum_{u \neq t}(x_u + 2)^2)$) and repeat the procedure until $x_i$ passes all the parity tests in the collection. In each scan of the collection, one new element of $G_i(x)$ is found and eliminated from the set $G_i(x)$; hence after $k$ such iterations all elements of $G_i(x)$ are found. One scan of the collection requires $O(k \log n)$ queries for finding a good subset and $O(\log n)$ queries for finding a larger element in this subset. Therefore, this scheme produces a non-constructive $O(k^2 \log n)$ algorithm for the problem.

We first observe that only $O(k \log n)$ queries are needed. This is because once the queries of the first phase are evaluated, they determine the results of the queries in subsequent phases. Specifically, let $S_1, \ldots, S_M$ be the subsets used. Let $b_1, \ldots, b_M$ be the results of the queries in the first phase, where $b_\ell = 1$ if $\prod_{j \in S_\ell \setminus \{i\}}(x_i - x_j) < 0$ and $b_\ell = 0$ otherwise. Assume $x_t$ is the element found in the first phase s.t. $x_t > x_i$. Then for each subset $S_\ell$ s.t. $t \in S_\ell$, the result of the query on $S_\ell$ in the second phase is $\neg b_\ell$ and it is $b_\ell$ otherwise. Generally, for $j > 1$ the results of the queries in the $j+1$ phase can be determined from the results of phase number $j$. Hence, we need to evaluate only $M + k \log n = O(k \log n)$ queries throughout the algorithm.

In order to obtain an explicit algorithm using this scheme, the collection of subsets should be constructed. We claim that this can be done using a $k$-wise $\epsilon$-biased probability space. Since $|G_i(x)| \leq k$, for any subset $U \subseteq \{1, \ldots, n\}$ at most $k$ elements in $U$ are larger than $x_i$, that is, $|U \cap G_i(x)| \leq k$.

A $k$-wise $\epsilon$-biased probability space with $n$ random variables that takes their values from $\{0, 1\}$ gives us even a stronger property than needed. Each point in a $k$-wise $\epsilon$-biased probability space represents a subset of $\{1, \ldots, n\}$, and for every $x$ with $1 < rank(x_i) \leq k+1$, a fraction $\delta \geq 1/2 - \epsilon$ of these subsets contain an odd number of elements larger than $x_i$. As was recalled in Section 4.2, a $k$-wise $\epsilon$-biased probability space of $n$ random variables can be constructed where the size of the space is $O(k \log n)$. Using the points of this space as the collection of subsets, we get an *explicit* $O(k \log n)$ algorithm that finds all members of $G_i(x)$, for every $x$ with $1 < rank(x_i) \leq k+1$.

The details of the procedure are as follows: $D$ will denote the set of larger elements found so far.

1. Initialize: $\epsilon \leftarrow \frac{1}{4}$; $D \leftarrow \emptyset$;
   Let $\{Q_1, \ldots, Q_M\}$ be sets represented by the random strings of a $k$-wise $\epsilon$-biased probability space as above. Thus $M = O(k \log n)$.
2. For $j = 1$ to $M$
   check if $Q_j$ contains an odd number of elements from $G_i(x)$:

$$b_j = \begin{cases} 1 & \text{if } \prod_{\ell \in Q_j \setminus \{i\}}(x_i - x_\ell) < 0 \\ 0 & \text{otherwise.} \end{cases}$$

3. Repeat until all elements of $G_i(x)$ are discovered.
   (a) For $j = 1$ to $M$
   check if $Q_j \setminus (\{i\} \bigcup D)$ contains an odd number of elements from $G_i(x)$:
   If in second phase or higher, then reverse $b_j$ if the last found element $t \in Q_j$.

(b) For the first $j$ s.t. $b_j = 1$, perform the following binary search on $Q_j$ to find $t \in Q_j$ s.t. $x_t > x_i$:
- Assign $V \leftarrow Q_j$.
- While $|V| > 1$: Let $U$ be the first $|V|/2$ elements of $V$. If $U$ contains an odd number of elements from $G_i(x)$ then set $V \leftarrow U$; otherwise $V \leftarrow |V| \backslash |U|$.
- Set $x_t \leftarrow$ the only element in $V$, $D \leftarrow D \bigcup \{t\}$.
- Replace $x_t$ by a very small element, say $-(\sum_{u \neq t}(x_u + 2)^2)$, and continue to step 2.

(c) If no new element was found, then terminate.

LEMMA 5.1. *For every $i$ with $1 < rank(i) \leq k + 1$ in $\{1, \ldots, n\}$, the above algorithm finds all the elements from $G_i(x)$, and the maximum of $x$, using $O(k \log n)$ parity tests.*

*Proof.* Suppose that at the beginning of iteration $M$ of the repeat loop, not all the elements of $G_i(x)$ were found yet, $1 < rank(x_i)$ in the current input, and all the input elements are distinct. By the property of a $k$-wise $\epsilon$-biased space, we are guaranteed that for some $1 \leq j \leq M$, $\prod_{\ell \in Q_j \backslash (\{i\} \bigcup D)}(x_i - x_\ell) < 0$ and we find a new $t \in G_i(x)$. After adding $\ell$ to $D$, the rank of $x_i$ is decreased by 1. Thus, as long as not all the elements of $G_i(x)$ were found, another iteration of the repeat loop will find a new element from $G_i(x)$. Thus, since $|G_i(x)| \leq k$, after at most $k$ iterations, all the elements of $G_i(x)$ are found. We perform at most $O(k \log n)$ parity tests in step 2 to get the results on all subsets. Additional $\log n$ parity tests are required at each phase of the repeat loop, for finding a new element from $G_i(x)$ in the good subset found. Thus, the total complexity is $O(k \log n)$. ■

Uehara *et al.* [UTW96] considered an analogous problem in the theory of attribute-efficient learning with $k$ essential attributes. They examined the computational complexity of learning the class of parity functions $PAR(k)$ defined as follows: Let $g$ be the parity function on $n$ variables $x_1, \ldots, x_n$. Denote by $g_S$ the sub-function of $g$ obtained from $g$ by replacing by 0 each $x_i$ in the input such that $i \notin S$. The class $PAR(k)$ contains all $g_S$ where $|S| = k$. Given a function $f$, the problem is to find $S$ s.t. $g_S = f$, that is, to find the $k$ essential variables. Note that the $k$ essential variables here correspond to the $k$ largest elements in our case since we use parity functions to find them.

They gave a non-constructive scheme to the problem of learning $PAR(k)$. In their model, adaptively chosen inputs $a \in \{0, 1\}^n$ are provided, and the correct value of $f(a)$ is given in response. An input $a = (a_1, \ldots, a_n)$ corresponds to the query set $A = \{i \mid a_i = 1\}$. If $f(a) = 1$, then an essential element can be found using a binary search similar to Ting's technique. They use the fact that we have implicit knowledge on a query set $S \backslash \{i\}$, once $S$ was asked and $i$ is an essential element. (We use this idea in our algorithm for finding the $k$ largest elements.) The scheme of [UTW96] gives a non-constructive $O(k \log n/k)$ upper bound for the problem (similar to the non-constructive bound we had in the preliminary version of the paper). Our current constructive bound of $O(k \log n)$ is applicable to the problem of [UTW96] as well.

## 6 FINDING THE MAXIMUM USING $O(\log^2 n)$ QUERIES

As mentioned before, there is a linear lower bound on the decisional complexity of deterministic, non-deterministic and Las Vegas algorithms for the problem of deciding maximality of an element in a set of $n$ *distinct* elements. Obviously, this also gives a lower bound on the decisional complexities of the corresponding search problem.

Ting and Yao [Ting93, TY94] presented a randomized algorithm that finds the maximum of $n$ distinct real numbers in the probabilistic polynomial decision tree model. For every constant $c > 0$ they presented an algorithm that uses $O(\log^2 n)$ polynomial queries and $O(n \log^2 n)$ random bits, and has error probability $O(1/n^c)$. At the general step of the algorithm they have a candidate element $x_i$ which they try to improve by applying a procedure which finds an element $x_{i'}$ larger than $x_i$. The algorithm is started with a uniformly randomly chosen $x_i$ and the above step is repeated until no such $x_{i'}$ is found, or a bound on the number of iterations is reached.

The procedure for finding a larger element $x_{i'}$ is as follows:

- Choose $O(\log n)$ independent uniform random subsets of $\{1, \ldots, n\} \backslash \{i\}$.

- Perform a parity test on each subset to find a good subset that contains and odd number of elements larger than $x_i$.
- Find a larger element $x_{i'}$ in the good subset by a binary search on the subset.

Ben-Or [BenOr96b] obtained an algorithm for the problem that does not require uniqueness of the input elements and has the same decisional and randomness complexities as the algorithm of [TY94]. The algorithm applies the procedure that was described in Section 4.3.1.

We present a different algorithm for finding the maximum of *n distinct* elements. Our motivation is to reduce the number of random bits so that both the decisional complexity and randomness complexity of the algorithm are $O(\log^2 n)$.

Since the algorithm of [TY94] is order invariant,[1] a probabilistic argument shows that even $O(\log n)$ random bits suffice for finding the maximum with $O(\log^2 n)$ polynomial queries and $O(1/n^c)$ error probability.

In the following sections we describe an explicit algorithm that finds the maximum using $O(\log^2 n)$ polynomial queries, $O(\log^2 n)$ random bits, and $O(1/n^c)$ probability of error.

## 61. Intuition of the Algorithm

Our algorithm uses a recursive procedure $Max(S)$ that with high probability returns the maximum of $\{x_i \mid i \in S\}$ where $S \subseteq \{1, \ldots, n\}$. The procedure chooses a subset $S_1 \subseteq S$, finds its maximum recursively, and then uses it to find the maximum of $S$. As in the algorithm of [TY94] we also have a candidate for a maximal element. In our algorithm it is the maximum of the subset $S_1$ chosen at the current recursive call. We use a different procedure for obtaining a larger element; namely, we find all the larger elements in the bigger set $S$ using the deterministic algorithm for finding the largest elements from Section 5. Another difference from the algorithm of [TY94] is that we chose all the subsets required in the algorithm over a $k$-wise $\delta$-dependent probability space instead of a uniformly random probability space.

The procedure goes as follows: if $1 \leq |S| \leq (c+1)\log n$ then find the maximum deterministically. For $|S| > (c+1)\log n$ we use the fact that if we choose a $k$-wise $\delta$-dependent random $S_1 \subseteq S$ where $k = (c+1)\log n$, then with high probability this subset contains an elements with a low rank in $S$. Find the maximum $x_z$ of $S_1$ recursively (with high probability). If $x_z$ is also the maximum of $S$, then we are done; otherwise we can apply the algorithm of Section 5 to find the elements larger than $x_z$ and also find the maximal of them in the process. Choosing the subset $S_1$ over a $k$-wise $\delta$-dependent probability space we gain several things: we still have the properties that a uniform probability space gives us, namely, with high probability, the size of $S_1$ is smaller by a constant factor than the size of the original subset $S$, and thus, after $O(\log n)$ recursive calls we will probably get to a subset of size $O(\log n)$. The second property of $S_1$ that resembles a uniformly chosen subset is that with high probability $S_1$ contains an element with a low rank, and thus, on the average, finding the elements larger than $x_z$ will take $O(\log n)$ parity tests, as described later. The advantages of choosing a $k$-wise $\delta$-dependent subset instead of a uniform one is that it much more economic in random bits, and enables us to achieve polylogarithmic randomness complexity.

## 62. Scheme of the Main Algorithm

For finding the maximum of $\{x_1, \ldots, x_n\}$ perform the following:

- Initialize $S = \{1, \ldots, n\}$,

$$\delta \leftarrow \frac{1}{n^{c+1}}; a \leftarrow 4(c+1); b \leftarrow \frac{c}{\log(\frac{2}{1+2\delta})};$$

$$m_1 \leftarrow \lceil b \log n \rceil + \lceil \log \frac{1}{\frac{1}{2}+\delta} n \rceil + 1;$$

- Return $Max(S, n)$.

[1] The queries used in the algorithm give the same answer on inputs $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ that satisfy for every $i, j x_i < x_j \Leftrightarrow y_i < y_j$.

The procedure $Max(S)$ returns the index of the maximum of $\{x_j \mid j \in S\}$, where $S \subseteq \{1, \ldots, n\}$, and goes as follows:
$Max(S, n)$:

1. If $|S| \leq (c+1)\log n$ then find the maximum of $S$ deterministically.
2. If more than $m_1$ recursive calls were performed then terminate.
3. Otherwise, for $k = (c+1)\log n$, choose a $k$-wise $\delta$-dependent subset $S_1$ of $S$ (if empty then terminate).
4. $z \leftarrow Max(S_1, n)$.
5. If $x_z$ is the maximum of $S$ then return $z$.
6. Otherwise, find the elements in $S$ larger than $x_z$ as follows:
   - Set $k \leftarrow 2$
   - While $k \leq (c+1)\log n$ and the maximum of $S$ war not found:

   (a) Apply the algorithm of Section 5 for $k$ to find the elements larger than $x_z$ and their maximum $x_\ell$.

   (b) If $k > 2$ and less than $k/2 + 1$ larger elements were found or $k = 2$ and no element was found, then conclude that the rank of $x_z$ is larger than $k+1$ and set $k \leftarrow 2k$.

   (c) Otherwise, test if $x_\ell$ is the maximum of $S$ by reusing the random walk of step 5. If $x_\ell$ passed the test, return $\ell$; otherwise set $k \leftarrow 2k$.
7. If $k > (c+1)\log n$ and the maximum of $S$ was found yet, then terminate.

Let $S = \{j_1, j_2, \ldots, j_t\} \subseteq \{1, \ldots, n\}$ and $x_{j1} < x_{j2} < \cdots < x_{jt}$. In order to choose a subset $S_1 \subseteq S$ in step 3, we set $k = (c+1)\log n$ and use $t$ random variables: $y_1, y_2, \ldots, y_t$ that take their values from $\{0, 1\}$ and are $k$-wise $\delta$-dependent. The subset $S_1$ is defined as $S_1 = \{j_i \mid y_i = 1\}$.

In step 5, checking whether $x_z$ is the maximum of $S$ is done by performing a random walk of length $a \log n$ on an expander graph, as described in Section 4.3.

We enter step 6 with an element $x_z$ that is known to have at least one element in $\{x_j \mid j \in S\}$ larger than it. The first iteration of the while loop finds all the elements larger than $x_z$, for $x_z$ with $rank(x_z) \leq 3$, and the maximum of these elements will pass the test on the random walk. Similarly, if the number of elements larger than $x_z$ is between $2^{j-1}+1$ and $2^j$, and $2 \leq j \leq \log(c+1)\log n$, then iteration $j$ finds all the larger elements.

The correctness of the algorithm is established in the next theorem.

THEOREM 6. *For any constant $c > 0$, the algorithm finds and maximum of $\{x_1, \ldots, x_n\}$ with error probability $O(1/n^c)$, using an expected number of $O(\log^2 n)$ parity tests and $O(\log^2 n)$ random bits.*

*Proof.* Define the events:

$A_1$: After $m_1$ recursive calls we still have a set $S$ with $|S| > (c+1)\log n$.

$A_2$: At most $m_1$ call were performed and an error occured at one of the $m_1$ calls.

It follows that

$$\Pr[error] \leq \Pr[A_1] + \Pr[A_2]\Pr[\neg A_1] \leq \Pr[A_1] + \Pr[A_2].$$

By [Ka91], Theorem 1, $\Pr[A_1] \leq 1/n^c$.

Assume $S = U_i$ at level $i$ of the recursion ($U_1 = \{1, \ldots, n\}$), and $z = z_i$ is the element returned at step 4 of this level. At level $i$ that is not the bottom level, an error may occur because of one of the following events:

$E_i$: $|U_i| > (c+1)\log n$ and none of the $(c+1)\log n$ largest elements in $U_i$ is in the subset $S_1 \subseteq U_i$ chosen at step 3.

$B_i$: an element that is not the maximum of $U_i$ passed the maximality check on the expander at step 5 or 6.

If $E_i$ occurs, then either $S_1$ us empty, and the algorithm terminates in step 3, or step 6 might not find all the elements larger than $x_{z_i}$. By the property of $k$-wise $\delta$-dependent random variables, for $k = (c+1)\log n$

$$\Pr[E_i] \leq \left(\frac{1}{2}\right)^{(c+1)\log n} + \delta = \frac{2}{n^{c+1}}.$$

The probability of error on one random walk is $O(1/n^c)$. The random walk check is executed for at most $1 + \log(c+1)\log n$ elements, and thus, $\Pr[B_i] \leq (\log((c+1)\log n) + 1)/(n^{c+1})$.

$$\Pr[A_2] \leq \sum_{i=1}^{m_1}(E_i + B_i) < \frac{1}{n^c}.$$

Hence the total error probability is at most $2/n^c$.

We show that the expected number of parity tests performed by the algorithm is $O(\log^2 n)$. Define the following notations:

$T$ = The number of parity tests performed during the execution of the algorithm on input $\{x_1, \ldots, x_n\}$.

$H$ = The number of recursive calls performed by the algorithm.

$L_i$ = The number of parity tests performed in step 5 at level $i$ of the recursion.

$M_i$ = The number of parity tests performed in step 6 at level $i$ of the recursion.

At the bottom level of the recursion at most $(c+1)\log n$ queries are performed, and thus,

$$T \leq (c+1)\log n + \sum_{i=1}^{H}(L_i + M_i)$$

$$\mathrm{E}[T] \leq (c+1)\log n + \sum_{i=1}^{m_1}(\mathrm{E}[L_i] + \mathrm{E}[M_i]).$$

$L_i \leq a \log n$, for every $1 \leq i \leq m_1$.

The rank of $z_i$ in $U_i$ is a random variable $r(z_i, U_i)$. If $r(z_i, U_i) = k$, then at step 6 we run the algorithm for finding the $m$ largest elements in $U_i$, for $m = 2, 4, \ldots, \min\{(c+1)\log n, 2^{\lceil \log k \rceil}\}$. Following each such run, we might need to perform a random walk of length $a \log n$. The number of parity tests performed for $x_{z_i}$ with rank $k$ s.t. $2^{\ell-1} + 1 \leq k \leq 2^\ell$ where $1 \leq \ell \leq \lceil \log((c+1)\log n) \rceil$ is at most $\sum_{j=1}^{\ell}(c_1 2^j \log n + a \log n) \leq a_1 2^\ell \log n$, for some constant $a_1 > 0$. If $k > (c+1)\log n$ then at most $O(\log^3 n)$ parity tests are performed. It follows that

$$\begin{aligned}\mathrm{E}[M_i] &= \mathrm{E}[\mathrm{E}[M_i \mid r(z_i, U_i) = k]] \\ &\leq \Pr[error] a_3 \log^2 n + a_2 \log n \Pr[1 \leq k \leq 2] \\ &\quad + a_1 \log n \sum_{j=1}^{\lceil \log(c+1)\log n \rceil} \Pr[2^{j-1} < k \leq 2^j] 2^j,\end{aligned}$$

for some constant $a_1, a_2, a_3$.

Define the event $B_i$: the rank of $x_{z_i}$ in $\{x_j \mid j \in U_i\}$ is $k+1$. By the property of $k$-wise $\delta$-dependent variables, $\Pr[B_i] \leq \frac{1}{2^{k+1}} + \delta$, and we have that for $2 \leq j \leq \lceil \log(c+1)\log n \rceil$

$$\Pr[2^{j-1} + 1 \leq k \leq 2^j] < \frac{1}{2}\frac{1}{2^{2^{j-1}}} + 2^{j-1}\delta$$

$$\Pr[k = 1 \vee k = 2] \leq \frac{3}{8} + 2\delta.$$

As was shown, $\Pr[error] \leq 2/n^c$. Substituting $\delta = 1/n^{c+1}$, we get that

$$\begin{aligned}\mathrm{E}[M_i] &\leq \frac{2}{n^c} a_3 \log^2 n + a_1 \left(\frac{3}{2} + 8\delta\right) \log n + a_1 \log n \sum_{j=2}^{\lceil \log(c+1) \log n \rceil} 2^j \left(\frac{1}{2^{j-1}} + 2^{j-1}\delta\right) \\ &= O(\log n)\end{aligned}$$

and finally, $\mathrm{E}[T] = O(\log^2 n)$.

It remains to show that the number of random bits required by the algorithm is $O(\log^2 n)$. At recursive level $i$ we look for the maximum of $U_i \subseteq \{1, \ldots, n\}$. Step 3 requires $O(\log |U_i| + \log(1/\delta)) = O(\log n)$ bits to choose $|U_i|$ $k$-wise $\delta$-dependent random variables. $O(\log n)$ random bits are required for checking maximality of an element in the set $U_i$ in step 5. At step 6c we reuse the random walk of step 5, and thus no more random bits are required. Since there are at most $O(\log n)$ recursive iterations, the total number of random bits required is $O(\log^2 n)$. ■

## 7 FURTHER RESEARCH

We saw that allowing $O(1/poly)$ probability of error can improve the running time of problems that have small co-non-deterministic complexity but high non-deterministic complexity.

The first open question that arises is how much further can the randomized decisional complexity of *finding* the maximum be reduced s.t. the error probability remains $O(1/poly)$.

It would also be interesting to extend the results of [TY94, MPR94, BenOr96b] to a broader class of problems, where the set at hand is a general semi-algebraic set, not necessarily represented by a conjunction of a finite number of inequalities.

## ACKNOWLEDGMENTS

We thank Michael Ben-Or for explaining his recent results.

## REFERENCES

[ABNNR92] Alon, N., Bruck, J., Naor, J., Naor, M., and Roth, R. (1992), Construction of asymptotically good low-rate error correcting codes through pseudo-random graphs, *IEEE Trans. Inform. Theory* **38**, 509–516.

[AGHP92] Alon, N., Goldreich, O., Hastad, J., and Peralta, R. (1992), Simple constructions for almost k-wise independent random variables, *Random Structures Algorithms* **3**, 289–304.

[AKS87] Ajtai, M., Komlos, J., and Szemeredi E. (1987), Deterministic simulation in LOGSPACE, *in* "Proceedings, 19th ACM Symposium on Theory of Computing," pp. 132–140.

[AS92] Alon, N., and Spencer, J. H. (1992), "The Probabilistic Methods," Wiley, New York.

[Ben Or83] Ben-Or, M. (1983), Lower bounds for algebraic computation trees, *in* "Proceedings, 19th ACM Symposium on Theory of Computing," pp. 80–86.

[Ben Or96a] Ben-Or, M. (1996), Randomized analytic decision trees—Summary of results, manuscript.

[Ben Or96b] Ben-Or, M. (1996), Finding the minimum with randomized polynomial tests, manuscript.

[BCR87] Bochnak, J., Coste, M., and Roy, M. F. (1987), "Géométrie algébrique réelle," Ergebnisse der Math. 3, Folge, Band 12, Springer-Verlag, Berlin.

[BKL93] Buergisser, P., Karpinski, M., and Lickteig, T. (1993), On randomized algebraic test complexity, *J. Complexity* **9**, 231–251.

[Chernoff52] Chernoff, H. (1952), A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, *Ann. Math. Statist.* **23**, 493–507.

[CW89] Cohen, A., and Wigderson, A. (1989), Dispersers, deterministic amplification, and weak random sources (extended abstract), *in* "Proceedings, 30th IEEE Symposium on Foundations of Computer Science," pp. 14–19.

[DL79] Dobkin, D. P., and Lipton, R. J. (1979), On the complexity of computations under varying sets of primitives, *J. Comput. System Sci.* **18**, 86–91.

[FRPU94] Feige, U., Peleg, D., Ragahavan, P., and Upfal, E. A. (1994), Computing with noisy information, *SIAM J. Comput.* **23**, 1001–1018.

[GKY95] Grigoriev, D., Karpinski, M., and Yao, A. C. (1995), An exponential lower bound on the size of algebraic decision trees, Technical Report TR-95-066, International Computer Science Institute, Berkeley.

[Gri99] Grigoriev, D. (1999), Randomized Complexity Lower Bound for Arrangements and Polyhedra, *Discrete Comput. Geom.* **21**, pp. 329–344.

[HR89] Hagerup, T., and Rub, C. (1989), A guided tour of Chernoff bounds, *Inform. Process. Lett.* **33**, 305–308.

[IZ89] Impagliazzo, R., and Zuckerman, D. (1989), How to recycle random bits, *in* "Proceedings, 30th IEEE Symposium on Foundations of Computer Science," pp. 248–253.

[Jar81] Jaromczyk, J. (1981), An extension of Rabin's complete proof concept, Symposium on mathematical foundations of computer science, *Lecture Notes Comput. Sci.* **118**, 321–326.

[Jo69] Jordan, C. (1969), Sur le assemblages des lignes, *J. Reine Angew. Math.* **70**, 185–190.

[Jus72] Justesen, J. (1972), A class of asymtotically good algebraic codes, *IEEE Trans. Inform. Theory* **18**, 652–656.

[Ka91] Karp, R. M. (1991), Probabilistic recurrence relations, *in* "Proceedings, 23rd ACM Symposium on Theory of Computing," pp. 190–197.

[LPS86] Lubotzky, A., Phillips, R., and Sarnak, P. (1986), Explicit expanders and the Ramanujan conjectures, *in* "Proceedings, 18th ACM Symposium on Theory of Computing," pp. 240–246.

[Mar88] Margulis, G. A. (1988), Explicit group-theoretical constructions of combinatorial schemes and their applications to the design of expanders and superconcentrators, *Problems Inform. Transmission* **24**, 39–46.

[Mey85a] Meyer Auf Der Heide, F. (1985), Simulating probabilistic by deterministic algebraic computation trees, *Theoret. Comput. Sci.* **3**, 325–330.

[Mey85b] Meyer Auf Der Heide, F. (1985), Nondeterministic versus probabilistic linear search algorithms, *in* "Proceedings 26th IEEE Symposium on Foundations of Computer Science," pp. 65–73.

[Meg83] Megiddo, N. (1983), Applying parallel computation algorithms in the design of serial algorithms, *J. Assoc. Comput. Mach.* **30**, 852–865.

[MPR94] Montana, J. L., Pardo, L. M., and Recio, T. (1994), A note on Rabin's width of a complete proof, *Comput. Complexity* **4**, 12–36.

[MT85] Manber, U., and Tompa, M. (1985), The complexity of problems on probabilistic, nondeterministic, and alternating decision trees, *J. Assoc. Comput. Mach.* **32**, 720–732.

[NA91] Naor, M. (1991), String matching with preprocessing of pattern and text. *in* "Proceedings of the 18th International Colloquium on Automata, Languages and Programming," Lecture Notes in Computer Science, Vol. 510, pp. 739–750, Springer Verlag, Berlin.

[Ni91] Nisan, E. (1991), CREW PRAMs and decision trees, *SIAM J. Comput.* **20**, 999–1007.

[NN93] Naor, J., and Naor, M. (1993), Small-biased probability spaces: Efficient constructions and applications, *SIAM J. Comput.* **22**(4), 838–856.

[Rab72] Rabin, M. O. (1972), Proving simultaneous positivity of linear forms, *J. Comput. System Sci.* **6**, 639–650.

[Ru95] Ruah, S. (1995), The decisional complexity of membership and selection problems over the reals, M.Sc. thesis, under the supervision of M. Naor, Weizmann Institute of Science.

[RY80] Rivest, R. L., and Yao, A. C. (1980), On the polyhedral decision problem, *SIAM J. Comput.* **9**, 343–347.

[S85] Snir, M. (1985), Lower bounds on probabilistic linear decision trees, *Theoret. Comput. Sci.* **38**, 69–82.

[SY82] Steele, J. M., and Yao, A. C. (1982), Lower Bounds for algebraic decision trees, *J. Algorithms* **3**, 1–8.

[Ting93] Ting, H. F. (1993), Computational complexity for selection problems with parity-like tests, Ph.D. thesis, Princeton University.

[TY94] Ting, H. F., and Yao, A. C. (1994), A randomized algorithm for finding maximum with $O((\log n)^2)$ polynomial tests, *Inform. Process, Lett.* **49**, 39–43.

[UTW96] Uehara, R., Tsuchida, K., and Wegener, I. (1997), Optimal attribute-efficient learning of disjunction, parity, and threshold functions, In Third European Conference on Computational Learning Theory (EuroCOLT '97), Lecture Notes in Artificial Intelligence, Vol. 1208, pp. 171–184, Springer-Verlag, Berlin.

[WY96] Widgerson, A., and Yao, A. (1996), A lower bound for finding the minimum on probabilistic decision trees with minimum tests, preprint.