Weak Random Sources, Hitting Sets, and BPP Simulations*

Alexander E. Andreev[†] Andrea E. F. Clementi[‡] José D. P. Rolim[§]
Luca Trevisan[¶]

January 26, 1998

Abstract

We show how to simulate any BPP algorithm in polynomial time using a weak random source of r bits and min-entropy r^γ for any $\gamma > 0$. This follows from a more general result about *sampling* with weak random sources. Our result matches an information-theoretic lower bound and solves a question that has been open for some years. The previous best results were a polynomial time simulation of RP [Saks, Srinivasan and Zhou 1995] and a quasi-polynomial time simulation of BPP [Ta-Shma 1996].

Departing significantly from previous related works, we do not use extractors; instead, we use the OR-disperser of [Saks, Srinivasan, and Zhou 1995] in combination with a tricky use of hitting sets borrowed from [Andreev, Clementi, and Rolim 1996].

AMS Subject Classification: 68Q10, 11K45.

Key Words and Phrases: Derandomization, Imperfect Sources of Randomness, Hitting Sets, Randomized Computations, Expander Graphs.

Abbreviated Title: BPP Simulations using Weak Random Sources.

1 Introduction

Randomized algorithms are often the simpler ones to solve a given problem, or the most efficient, or both (see [MR95]). For some problems, including primality testing and approximation of # P-complete counting problems, only randomized solutions are known.

The practical applicability of such randomized methods depends on the effective possibility for an algorithm to access *truly random bits*. Since it is questionable whether truly random sources really exist, much research has been devoted in the last decade to find weaker notions of randomness that are still sufficient to run BPP algorithms in polynomial time [VV85, SV86, Vaz86, Vaz87, CG88, Zuc90]. Several definitions of *weak random source* have been proposed in the literature, the most general being the following [CG88, Zuc90]: for $\gamma > 0$, an (r, r^γ)-source is a random source that

*An extended abstract of this paper appears in the Proceedings of the *38-th IEEE Symposium on Foundations of Computer Science.*

†andreev@mntn.msk.su. University of Moscow.
‡clementi@dsi.uniroma1.it. University of Rome *La Sapienza.*
§rolim@cui.unige.ch. University of Geneva.
¶trevisan@theory.lcs.mit.edu. MIT.

outputs a string in $\{0,1\}^r$ and no string has probability of being output larger than 2^{-r^γ} (such object is also called *random source of min-entropy* r^γ). An information-theoretic argument shows that a black-box simulation of BPP using an $(r, r^{o(1)})$-source is impossible when r is polynomial in the number of random bits used by the simulated algorithm.

Dispersers and Extractors

The usual method to simulate a BPP algorithm using a weak random source is as follows. Say that, for a given input, the algorithm requires m (truly) random bits; then we ask the source r bits (note that it is required to make only one access to the weak random) and we use them to produce a sample space (a set of m-bit strings). Such strings are fed into the algorithm and then the majority rule is used to decide whether to accept or reject. The procedure that computes the sample space starting from the output of the source is *independent* of the algorithm that we want to derandomize. This simulation is basically equivalent [Zuc90, Zuc96b, NZ96, SZ94, SSZ95, TS96] to a bipartite graph $G = (V, W, E)$ having 2^r nodes in the left component V, 2^m nodes in the right component W, degree d and such that if we select a node v in the left component according to an (r, r^γ)-source, and then a random neighbour of v, the induced distribution in W is ϵ-close to the uniform distribution over W. Such graph is a $(2^r, 2^m, d, r^\gamma, \epsilon)$-*extractor*. The left nodes are seen as possible outcomes of the random source, the right nodes as possible random strings for the algorithm to be simulated. The simulation amounts to select a node in the left side according to the weak random source and then select as sample space the set of its neighbours. If, for some fixed γ, one could achieve d and r polynomial in m, then a polynomial time simulation of BPP would be possible using an (r, r^γ)-source. However, the best present construction of extractors for fixed $\gamma > 0$ and $r = \text{poly}(m)$ has $d = n^{log^{(k)} n}$ [TS96]. This implies a quasi-polynomial time simulation of BPP. A polynomial-time simulation of BPP using weak random sources of min-entropy r^γ for any fixed $\gamma > 0$ was one of the major open questions in the field.

It is not difficult to show that to simulate RP by means of a weak random source, *OR dispersers* [CW89] (from now on, we will simply call them *dispersers*) are sufficient. A $(2^r, 2^m, 2^{r^\gamma})$-disperser is again a bipartite graph $G = (V, W, E)$ with parameters r, m, and d as before, but now the property is that for any set $V' \subseteq V$ of at least 2^{r^γ} vertices on the left side and any set $W' \subseteq W$ of more than $2^m/2$ vertices on the right side, there is at least one edge joining V' and W'. This construction is somewhat easier to obtain, and Saks et al. [SSZ95] give indeed a disperser with $d = \text{poly}(n)$, for any constant $\gamma > 0$, allowing for a polynomial time simulation of RP.

See [Nis96] for a complete survey on extractors, dispersers, and weak random sources.

Pseudorandom Generators and Hitting Sets

A more ambitious goal than simulating BPP with weak random sources is the *deterministic* simulation of BPP. Research on this subject tries to isolate reasonable complexity assumptions under which deterministic simulations of randomized algorithms are possible [Y82, BM84, Nis90, BFNW93, NW94, IW97, ACR97b].

In some cases, combinatorial objects developed in the study of weak random sources have been used to give derandomization [NZ96]. Here we revert this connection, and use a derandomization method to take full advantage from a weak random source.

Two basic combinatorial objects are studied in the theory of derandomization: pseudorandom generators (whose efficient construction immediately implies a deterministic simulation of BPP) and hitting set generators (whose efficient construction allows to simulate RP algorithms). Informally speaking, in the context of derandomization, pseudorandom generators play the same role of extractors and hitting sets generators play that of dispersers. A recent result of Andreev et al. [ACR97a] shows how to deterministically simulate BPP algorithms using hitting set generators. This suggests that perhaps dispersers could be used to simulate BPP with weak random sources.

A *quick δ-hitting set generator* (quick δ-HSG) is an algorithm that, given a parameter n, finds in poly(n) time a set $H_n \subseteq \{0,1\}^*$ such that, for any finite Boolean function f of circuit complexity n, if $\mathbf{Pr}_x[f(x)=1] > \delta$ then $f(a)=1$ for some $a \in H_n$[1], where the probability is taken uniformly over $\{0,1\}^n$. The main technical result of [ACR97a] can be stated as follows.

Lemma 1 ([ACR97a]) *For any choice of constants $\epsilon, \delta > 0$, there is a deterministic algorithm that, given access to a quick δ-HSG, and given in input any circuit C of size n returns in* poly(n) *time a value D such that $|\mathbf{Pr}_x[C(x)=1] - D| \leq \epsilon$, where the probability is taken uniformly over all possible $x \in \{0,1\}^n$.*

Lemma 1 immediately implies the following general derandomization result.

Theorem 2 *If for some $\delta > 0$ a quick δ-HSG exists, then* P = BPP.

Andreev et al. [ACR97a] prove Lemma 1 by constructing a set S of size poly(n) that is ϵ-discrepant for C, i.e. such that $\mathbf{Pr}_{x\in S}[C(x)=1]$ approximates the value $\mathbf{Pr}_x[C(x)=1]$ up to an additive error ϵ. A basic ingredient is the definition of a *discrepancy test* that, given a circuit C, a "candidate" set S and a parameter ϵ, tests whether S is ϵ-discrepant for C. The test also needs an auxiliary set H in input, and, provided that H has a certain hitting property, the test is "sound", that is if the set S is accepted, then S is ϵ-discrepant for C. The fact that the test is sound only if the auxiliary set H is hitting is not a major restriction — since we are assuming that a hitting set generator exists, we can use it to generate H. Thus, proving the theorem amounts to find a set S that passes the test. This task is solved in [ACR97a] by means of a rather involved (and inherently sequential) algorithm. The algorithm indeed proves a somewhat stronger result than Lemma 1 and has been also used in [ACR97b] in a different context. For the sake of proving Lemma 1 it might however be over-kill.

Our Results

We show how to use *dispersers* and weak random sources to simulate BPP in polynomial time and to even solve a more general *sampling* problem.

The sampling problem we are interested in is as follows: Given oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ and a weak source of randomness, we want to find a set S of size poly(n) that with high probability is ϵ-discrepant for f. It should be clear that simulating a given BPP algorithm reduces to the above problem: the computation of a BPP algorithm on a fixed input is an (easy to compute) function f of the outcomes of the random coins. Being able to approximate the fraction of random coin outcomes that make f accept, allows to decide whether the algorithm accepts or not the input.

We show that dispersers are sufficient for the above sampling problem. The starting point is the observation that using a disperser and a weak random source it is possible to generate polynomially many small sets $S_1, \ldots, S_k$ and $H_1, \ldots, H_k$ such that, with high probability, one of the S_i's is ϵ-discrepant for f, and one of the H_i's has the hitting property required by the discrepancy test (see Theorem 21). Then, we define $H = \bigcup H_i$. Since the hitting property is *monotone* (adding elements to a set cannot decrease its hitting properties), we have that H will be a hitting set with high

[1] In the next section we will give a seemingly weaker (but in fact equivalent) formal definition.

probability. We can thus run the discrepancy test on the sets $S_1, \ldots, S_k$ using H as the reference hitting set. We shall then prove that, with high probability, one of the S_i's will pass the test and thus be ϵ-discrepant for f as required.

The main difference between our method and the extractor-based one (mentioned in the beginning) is that the ϵ-discrepant set that is given in output *depends on* the specific function f that is accessed as oracle. The source of this non-obliviousness is the selection of a good set S_j among the candidates $S_1, \ldots, S_k$. As a result, our sampling algorithm is not *oblivious* according to the definition of Bellare and Rompel [BR94], however it is *non-adaptive*. See [G97] for definitions of these notions and for a survey on sampling.

Our main result can be stated in the following way

Theorem 3 (Main Theorem) *For any $\gamma > 0$, there exist a polynomial p and a deterministic algorithm A such that the following holds. For any $\epsilon > 0$, $n > 0$, any $(p(n/\epsilon), p(n/\epsilon)^\gamma)$-source X, and any $f : \{0,1\}^n \to \{0,1\}$, on input (ϵ, n, X) and oracle access to f, A computes, in time polynomial in n/ϵ, a value D such that with probability at least $1 - 2^{-\mathrm{poly}(n)}$ over the outcomes of the source,*

$$|\mathbf{Pr}_x[f(x) = 1] - D| \leq \epsilon \ .$$

Note that since the algorithm runs in polynomial time it will make $\mathrm{poly}(n/\epsilon)$ queries to f.

Corollary 4 *For any $\gamma > 0$, any* BPP *algorithm can be simulated in polynomial time using an (r, r^γ)-source.*

The idea of generating candidate discrepancy sets $S_1, \ldots, S_k$ and then applying the discrepancy test to them also yields a simple proof of Lemma 1. This simplified proof is presented in a preliminary version of this paper [ACRT97] and also in an appendix of the final version of the paper of Andreev et al. [ACR97b]. More recently, Lance Fortnow has observed that an even simpler proof *of Theorem 2* can be given by using a previous result of Lautemann [L83]. Fortnow's proof of Theorem 2 does not use the discrepancy test. To the best of our understanding, this new proof does not extend to the context of dispersers and weak random sources, and it seems that we still need the discrepancy test in order to prove Theorem 3. An additional, and fairly surprising result observed by Fortnow is that BPP can be simulated by an RP machine having oracle access to a promise-RP problem. We present both Fortnow's results in Section 5.

Overview of the Paper

We give some definitions in Section 2. In Section 3 we describe the discrepancy test and its properties. In Section 4 we prove Theorem 3. Fortnow's proof of Theorem 2 is presented in Section 5. Section 6 is devoted to some concluding remarks.

2 Preliminaries

Unless otherwise stated, probabilities are with respect to the uniform distribution. For any positive integer n we denote by $\mathcal{F}_n$ the set of all n-ary Boolean functions $f : \{0,1\}^n \to \{0,1\}$. For a vector $a \in \{0,1\}^n$, and a function $f : \{0,1\}^n \to \{0,1\}$, we define a function $f^{\oplus a} : \{0,1\}^n \to \{0,1\}$ as $f^{\oplus a}(x) = f(x \oplus a)$.

We say that a Boolean function f *accepts* x if $f(x) = 1$.

Definition 5 (Weak random source) *A probability distribution D over the set $\{0,1\}^r$ is an* (r, r^γ)-source (weak random source of min entropy r^γ) *if for any $x \in \{0,1\}^r$, $D(x) \leq 2^{-r^\gamma}$.*

For a vertex v of a graph $G = (V, E)$ we let $\Gamma(v) \subseteq V$ be the set of vertices that are adjacent to v. For a subset $S \subseteq V$, we define $\Gamma(S) = \bigcup_{v \in S} \Gamma(v)$. We give here a definition of dispersers which is more convenient than that given in the Introduction to describe our results. It is easy to verify that the two definitions are in fact equivalent.

Definition 6 (Disperser) *A bipartite multigraph $G(V, W, E)$ with $|V| = R$ and $|W| = N$ is said to be an (R, N, T)-disperser if for any subset $S \subseteq V$ such that $|S| \geq T$, it holds $\Gamma(S) \geq N/2$.*

Definition 7 (Circuit complexity) *For a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ we denote by $L(f)$ the minimum size of a circuit computing f (here, for* circuit *we mean* a circuit whose gates have fan-in at most 2 and arbitrary fan-out.*)*

Definition 8 (Kolmogorov Complexity) *Let us fix a universal Turing machine U with alphabet $\{0,1\}$ for programs allowing oracle queries. Given two Boolean functions $f : \{0,1\}^k \rightarrow \{0,1\}$ and $g : \{0,1\}^n \rightarrow \{0,1\}$, we define the* conditional Kolmogorov complexity of g given f*, denoted $K_U(g|f)$, as the length of the shortest program for U that evaluates g having oracle access to f.*

For example, $K_U(f|f) = O(1)$. As usual, if we fix another universal Turing machine U' it holds $K_{U'}(g|f) = K_U(g|f) + \Theta(1)$. We will usually omit the subscript. See e.g. [LV90] for an introduction to Kolmogorov complexity. In this paper we only use the obvious fact that, for any fixed f, the number of functions g such that $K(g|f) \leq k$ is at most 2^k.

Definition 9 (Hitting set) *A (multi)set $H \subseteq \{0,1\}^n$ is said to be δ-hitting for a family of functions $\mathcal{G} \subseteq \mathcal{F}_n$ if for any $f \in \mathcal{G}$ with $\mathbf{Pr}_x[f(x) = 1] > \delta$ there exists $x \in H$ such that $f(x) = 1$.*

Recall that by our convention $\mathbf{Pr}_x(\cdot) = \mathbf{Pr}_{x \in \{0,1\}^n}(\cdot)$.

Definition 10 (Discrepancy set) *A (multi)set $S \subseteq \{0,1\}^n$ is said to be ϵ-discrepant for a family of functions $\mathcal{G} \subseteq \mathcal{F}_n$ if for any $f \in \mathcal{G}$,*

$$|\mathbf{Pr}_{x \in S}[f(x) = 1] - \mathbf{Pr}_x[f(x) = 1]| \leq \epsilon \ .$$

Note that if a set is ϵ-discrepant for a family $\mathcal{G}$ then it is also ϵ-hitting for $\mathcal{G}$, but the converse is not necessarily true.

The definition below is a slight variant of the definition of *quick δ-HSG of price* $O(\log n)$ given in [ACR97a].

Definition 11 (Hitting Set Generator) *A quick δ-HSG is a polynomial-time algorithm $\mathcal{H}$ that, on input a number n in unary, returns a multiset $\mathcal{H}(n) \subseteq \{0,1\}^n$ that is δ-hitting for the set $\{f : \{0,1\}^n \rightarrow \{0,1\} : L(f) \leq n\}$.*

It may seem awkward to restrict the above definition to functions having circuit complexity equal to the number of inputs. However any n-ary function of circuit complexity N can be seen as a N-ary function of circuit complexity N whose value is independent of $N - n$ of its inputs (this point of view does not change the fraction of satisfying inputs as long as we consider constant fractions as done below). As a consequence of this observation, the set $\mathcal{H}(n)$ returned by the HSG hits *any* function of circuit complexity at most n.

Using straightforward amplification, it is easy to show the following useful property of HSG's.

Lemma 12 ([ACR97a]) *let $\delta(n)$ and $k(n)$ be polynomial-time computable functions such that $0 < \delta(n) < 1$ and $k(n) \leq \text{poly}(n)$. Then if a quick $(1-\delta(n))$-HSG exists then there exists a quick $\left(1 - (\delta((k(n)+1)\cdot n))^{1/k(n)}\right)$-HSG. In particular, for any two constants $0 < \delta, \delta' < 1$, if there exists a quick δ-HSG, then there exists a quick δ'-HSG.*

PROOF: We use the standard *sequential repetition* method. For an input n, $\mathcal{H}'$ first computes (using $\mathcal{H}$) a set $H \subseteq \{0,1\}^{(k(n)+1)\cdot n}$ that is $(1-\delta((k(n)+1)\cdot n))$-hitting for all the functions $g : \{0,1\}^{(k(n)+1)n} \to \{0,1\}$ such that $L(g) \leq (k(n)+1)n$. Then, it generates a set $H' \subseteq \{0,1\}^n$ by "parsing" each element of H into $k(n)+1$ strings of length n.

We claim that H' is $\left(1 - (\delta((k(n)+1)\cdot n))^{1/k(n)}\right)$-hitting for functions of circuit size n. Let $f : \{0,1\}^n \to \{0,1\}$ be such that $L(f) \leq n$ and

$$\mathbf{Pr}_x[f(x)=1] \;>\; 1 - (\delta((k(n)+1)\cdot n))^{1/k(n)} \;.$$

Let $f^k : \{0,1\}^{(k(n)+1)n} \to \{0,1\}$ be the function that takes $k(n)+1$ strings of $\{0,1\}^n$ and whose value is 1 if and only if f evaluates to one on at least one of the first $k(n)$ strings. Note that $L(f^k) \leq k(n)L(f) + k(n) = k(n)\cdot(n+1)$. We have

$$\mathbf{Pr}_y[f^k(y)=1] \;=\; 1 - (\mathbf{Pr}_x[f(x)=0])^{k(n)} \;>\; 1 - \delta((k(n)+1)\cdot n) \;.$$

Due to its hitting property, H contains an input that satisfies f^k and thus H' contains an input that satisfies f. The main claim follows.

For the second claim, if $\delta' \geq \delta$ then there is nothing to prove since, by definition, a δ-HSG is also a δ'-HSG for any $\delta' \geq \delta$. If $\delta' < \delta$, then we take a large enough k such that $\delta' \geq (1-(1-\delta)^{1/k})$ and then we use the main claim. □

Observe that by applying the above Lemma with $k(n) = \text{poly}(n)$, it is possible to show that, for any $0 < \epsilon < 1$, the existence of a quick $(1-2^{-n^{1-\epsilon}})$-HSG implies to the existence of a quick $(1/\text{poly}(n))$-HSG. By using random walks on expander graphs instead that simple repetition, Andreev et al. [ACR97b] show that, for $c > 1/2$, even the existence of a $(1-2^{-cn})$-HSG is an equivalent condition.

3 The Discrepancy Test

In this section we describe the discrepancy test of Andreev et al. [ACR97a]. We present a slight variation of the proof of [ACR97a] that the test is sound, and also prove a "completeness" property of the test.

For any vector $a \in \{0,1\}^n$, function $f : \{0,1\}^n \to \{0,1\}$, and set $S \subseteq \{0,1\}^n$, define

$$p(a,f,S) = \mathbf{Pr}_{x\in S}[f(x\oplus a)=1] \;. \tag{1}$$

For any two subsets $S, H \subseteq \{0,1\}^n$, constant $\epsilon > 0$, and function $f : \{0,1\}^n \to \{0,1\}$, we define in Figure 1 a discrepancy test, denoted DISC-TEST(f,S,H,ϵ). In this test, the set S is tested to be ϵ-discrepant for f by using the auxiliary (hitting) set H.

Theorem 13 (Soundness of DISC-TEST [ACR97a]) *A constant c_1 exists such that, for any $\epsilon > 0$, integer n, function $f : \{0,1\}^n \to \{0,1\}$, sets $S, H \subseteq \{0,1\}^n$, if DISC-TEST$(f,S,H,\epsilon) = 1$ and H is δ-hitting for the set of functions g such that $K(g|f) \leq c_1 \cdot |S| \cdot n$, then S is $(\epsilon+\delta)$-discrepant for f.*

```
DISC-TEST(f, S, H, ε)
begin
  p_min := min{p(a, f, S) : a ∈ H ∪ {0⃗}};
  p_max := max{p(a, f, S) : a ∈ H ∪ {0⃗}};
  if p_max − p_min ≤ ε then return (1)
                        else return (0)
end
```

Figure 1: The discrepancy test.

Theorem 13 is the core of the results of [ACR97a]. Note that it says that a set H with a certain *one-sided* pseudorandom property (the hitting property) can be used to test S for a *two-sided* pseudorandom property (the discrepancy property). However H has to be hitting for *a whole set* of functions while S is tested for discrepancy *on a single function* (i.e. f). So, roughly speaking, the theorem trades-off "globality" versus "two-sidedness". The version of Theorem 13 proved in [ACR97a] requires f to be computable by a small circuit and H to be δ-hitting for a family of functions of low circuit complexity. Here we have no requirement on f and H is required to be hitting for a set of functions directly "related" to f.

PROOF: [Of Theorem 13] Let f, $S = \{s_1, \ldots, s_m\}$, H, ϵ be fixed throughout the proof, and suppose H is δ-hitting for all g's with $K(g|f) \leq c_1 mn$. Let us define the function BAD : $\{0,1\}^n \to \{0,1\}$ as follows

$$\text{BAD}_{f,S,H}(a) \stackrel{\text{def}}{=} \text{BAD}(a) = \begin{cases} 0 & \text{if } p_{\min} \leq p(a,f,S) \leq p_{\max}; \\ 1 & \text{otherwise.} \end{cases}$$

where $p_{\min}$ and $p_{\max}$ are as defined in Figure 1.

Claim 14 $K(\text{BAD}|f) \leq mn + 2\log m + O(1)$.

PROOF: We observe that BAD can be computed with the pseudo-code depicted in Figure 2.

Let us bound the length of such a program. All the elements of S have to be defined explicitly, and this can be done using mn bits; $p_{\min}$ and $p_{\max}$ have to be defined too, and since they are integral multiples of $1/m$, $\log m$ bits are sufficient to encode each of them. The rest of the program has constant length. We can conclude that the total length of the program is $mn + 2\log m + O(1)$. □

We fix c_1 large enough so that, using the hypothesis of the theorem, H is δ-hitting for BAD.

Claim 15 $\mathbf{Pr}_{a\in\{0,1\}^n}[\text{BAD}(a) = 1] \leq \delta$.

PROOF: Assume, by contradiction, that $\mathbf{Pr}_{a\in\{0,1\}^n}[\text{BAD}(a) = 1] > \delta$, then by the hitting property of H, there exists some $a \in H$ such that $\text{BAD}(a) = 1$, which is impossible by definition of BAD, $p_{\min}$ and $p_{\max}$ (as for any $a \in H$, we have $p_{\min} \leq p(a,f,S) \leq p_{\max}$). □

Let $\mathbf{E}[p(a,f,S)]$ be the average of $p(a,f,S)$ over all the choices of $a \in \{0,1\}^n$.

Claim 16 $\mathbf{E}[p(x,f,S)] = \mathbf{Pr}_x[f(x) = 1]$.

```
function BAD(a)
constants
   p_min, p_max, m;
   s_1, ..., s_m;
begin
   count := 0;
   for i := 1 to m do
      count := count + f(a ⊕ s_i);
   if count > m p_max or count < m p_min then
      return (1)
   else
      return (0)
end.
```

Figure 2: How to compute BAD. The algorithm has oracle access to f. It first computes the number of 1's of $f(a \oplus s_i)$ for $i = 1, \ldots, m$, and then decides whether to accept or reject by comparing this number with $p_{\min}$ and $p_{\max}$.

PROOF:

$$\begin{aligned}
\mathbf{E}\left[p(x,f,S)\right] &= \mathbf{Pr}_{x\in\{0,1\}^n\ ,\ y\in S}[f(x\oplus y)=1] \\
&= \frac{1}{|S|}\sum_{y\in S}\mathbf{Pr}_x[f(x\oplus y)=1] \\
&= \frac{1}{|S|}\sum_{y\in S}\mathbf{Pr}_x[f(x)=1] \\
&= \mathbf{Pr}_x[f(x)=1]\ .
\end{aligned}$$

□

From Claim 15 we have the following inequalities (where the first term is due to a's for which $\textsc{bad}(a) = 0$ and the second term is due to the rest):

$$\mathbf{E}\left[p(a,f,S)\right] \le (1-\delta)\cdot p_{\max} + \delta\cdot 1 \le p_{\max} + \delta \tag{2}$$

$$\mathbf{E}\left[p(a,f,S)\right] \ge (1-\delta)\cdot p_{\min} \ge p_{\min} - \delta \tag{3}$$

Recall that whenever the test accepts, $p_{\max} - p_{\min} \le \epsilon$. Also, by definition,

$$p_{\min} \le p(0,f,S) = \mathbf{Pr}_{x\in S}[f(x)=1] \le p_{\max}\ .$$

By Claim 16 and Eq. 2, we obtain

$$\mathbf{Pr}_x[f(x)=1] - \mathbf{Pr}_{x\in S}[f(x)=1] \le (p_{\max}+\delta) - p_{\min} \le \epsilon + \delta$$

and, similarly,

$$\mathbf{Pr}_{x\in S}[f(x)=1] - \mathbf{Pr}_x[f(x)=1] \le p_{\max} - (p_{\min}-\delta) \le \epsilon + \delta\ .$$

Thus, S is indeed $(\epsilon + \delta)$-discrepant for f, and Theorem 13 follows. □

We now give a sufficient condition for DISC-TEST to accept.

Theorem 17 (Completeness of DISC-TEST) *If S is $(\epsilon/2)$-discrepant for the set $\{f^{\oplus a} : a \in \{0,1\}^n\}$, then DISC-TEST$(f, S, H, \epsilon) = 1$, for any set $H \subseteq \{0,1\}^n$.*

PROOF: Fix $H \subseteq \{0,1\}^n$ and let a_1 (respectively, a_2) be a point where $p_{\min} = p(a_1, f, S)$ (resp. $p_{\max} = p(a_2, f, S)$).

$$\begin{aligned} p_{\min} &= \mathbf{Pr}_{x \in S}[f^{\oplus a_1}(x) = 1] \\ &\geq \mathbf{Pr}_x[f^{\oplus a_1}(x) = 1] - \epsilon/2 \\ &= \mathbf{Pr}_x[f(x) = 1] - \epsilon/2 \ . \end{aligned}$$

Where the first inequality is due to the discrepancy property of S. Similarly, we have

$$\begin{aligned} p_{\max} &= \mathbf{Pr}_{x \in S}[f^{\oplus a_2}(x) = 1] \\ &\leq \mathbf{Pr}_x[f^{\oplus a_2}(x) = 1] + \epsilon/2 \\ &= \mathbf{Pr}_x[f(x) = 1] + \epsilon/2 \ . \end{aligned}$$

and thus $p_{\max} - p_{\min} \leq \epsilon$. □

4 Proof of Theorem 3

The starting point of our proof is the following easy observation: If we have a set $I \subseteq \{0,1\}^N$ such that $\mathbf{Pr}_x[x \in I] > 1/2$, then using a weak random source and the dispersers of [SSZ95], we can generate a polynomial-sized (in N) set of vectors $x_1, \ldots, x_k$ such that, with high probability, $\{x_1, \ldots, x_k\} \cap I \neq \emptyset$. This is formalized in Corollary 19 below.

A naive way of using this fact would be to take the set I as the family of ϵ-discrepant sets S for f of size m. For large enough m ($m = O(1/\epsilon^2)$ would suffice) the set I will be such that $\mathbf{Pr}_{S \subseteq \{0,1\}^m}[S \in I] > 1/2$ and so we can use the weak random source and the disperser to generate a family of sets $S_1, \ldots, S_k$ such that, with high probability, one of them is ϵ-discrepant for f. But now the problem is that we are not able to recognize which of these sets has the discrepancy property (note that an efficient Las Vegas algorithm to test the discrepancy property would imply ZPP = BPP). Theorem 13 gives indeed a way to test for discrepancy, provided that we have a hitting set at hand.

We thus define $I \subseteq \{0,1\}^{(m+M)\cdot n}$ as the family of pairs of sets (H, S) such that H has M elements and the hitting property as in the hypothesis of Theorem 13 and S has m elements and the discrepancy property as in the hypothesis of Theorem 17. As shown in Lemma 20 below, for an appropriate choice of m and M, the set I is such that $\mathbf{Pr}_{(H,S)}[(H, S) \in I] > 1/2$. Using the weak random source we can thus obtain a set of pairs $(H_1, S_1), \ldots, (H_k, S_k)$ such that, for some j, the set S_j has the required discrepancy property and H_j the required hitting property (with high probability). The next important observation is that the hitting property is *monotone*, that is, if a set H has a certain hitting property, and J is any set, then $H \cup J$ has at least the same hitting property of H (the reader may note that the discrepancy property is *not* monotone). As a consequence, the set $\bigcup_i H_i$ has (with high probability) the hitting property required by Theorem 13.

We start by quoting the disperser construction of Saks, Srinivasan, and Zhou.

Theorem 18 (Construction of dispersers [SSZ95]) *For any $0 < \lambda < \alpha \leq 1$, for any sufficiently large r, and for any $2^{r^\alpha} \leq T \leq 2^r$, there exists an efficient construction of a $(2^r, 2^{r^\lambda}, T)$-disperser $G = (V, W, E)$ of degree* $\mathrm{poly}(r)$.

In the previous theorem, by "efficient construction" we mean the existence of an algorithm that for any vertex of V finds its neighbours in time $\mathrm{poly}(r)$.

Corollary 19 *For any choice of constants $0 < \gamma < 1$ and $c > 0$ there exist a polynomial p and an algorithm A such that the following property holds. For any $n > 0$, any set $I \subseteq \{0,1\}^n$ with $|I| > 2^{n-1}$, and any $(p(n), p(n)^\gamma)$-source X, algorithm A, on input (n, X), outputs a set $C \subseteq \{0,1\}^n$ of size* $\mathrm{poly}(n^{c/\gamma})$ *such that*

$$\mathbf{Pr}[C \cap I = \emptyset] \leq 2^{-n^c}$$

where the probability is taken over the outcomes of the source.

We will use Corollary 19 by taking I as the set of pairs (S, H) such that S has a certain discrepancy property and H has a certain hitting property. Observe that algorithm A computes the set of "candidates" C without "knowing" which set I has been fixed.

PROOF: [Of Corollary 19] Fix constants α and λ such that $0 < \lambda < \alpha < \gamma$ and $n^{c\lambda} \leq n^\gamma - n^\alpha$. Let $r = n^{1/\lambda}$. Consider a $(2^r, 2^n, 2^{r^\alpha})$-disperser $G = (V, W, E)$, that can be efficiently constructed as in Theorem 18. We identify V with $\{0,1\}^r$ and W with $\{0,1\}^n$. Let $B \subseteq V$ be the set of "bad" vertices v such that $\Gamma(v) \subseteq W - I$. We claim that $|B| < 2^{r^\alpha}$: otherwise we reach a contradiction since, by definition of disperser, we would have $|\Gamma(B)| \geq 2^n/2$, while $|W - I| < 2^n/2$. Let us select an element v of V using an (r, r^γ)-source, and let C be the set of its neighbours. The probability that we picked a bad vertex $v \in B$ is at most $2^{r^\alpha} \cdot 2^{-r^\gamma} = 2^{n^{\alpha/\lambda} - n^{\gamma/\lambda}} \leq 2^{-n^c}$. On the other hand, if $v \notin B$, then $C \cap I \neq \emptyset$; the corollary thus follows. □

As a preparation to using Corollary 19, we show that, for a randomly chosen pair of sets (S, H) of sufficiently large sizes, with high probability S has a certain discrepancy property and H has a certain hitting property.

Lemma 20 *There exists two constants c_2 and c_3 such that for any $\epsilon > 0$, $n > 0$, $f : \{0,1\}^n \to \{0,1\}$, $c > 0$ and for $m = c_2 n/\epsilon^2$ and $M = c_3 cmn/\epsilon$, for a randomly chosen element (v, u) (where $v \in \{0,1\}^{Mn}$ and $u \in \{0,1\}^{mn}$) the following holds with probability larger than $1/2$:*

1. *v, regarded as a multiset of $\{0,1\}^n$ of size M, is $\epsilon/2$-hitting for the set of functions g such that $K(g|f) \leq cmn$*
2. *u, regarded as a multiset of $\{0,1\}^n$ of size m, form a set that is $\epsilon/4$-discrepant for $\{f^{\oplus a} : a \in \{0,1\}^n\}$.*

PROOF: It suffices to prove that each event holds with probability larger than $3/4$.

Regarding the first event, the number of functions $g \in \mathcal{F}_n$ such that $K(g|f) \leq cmn$ is clearly at most 2^{cmn}. If one such g has $\mathbf{Pr}_x[g(x) = 1] \geq \epsilon/2$, then the probability that M randomly chosen elements from $\{0,1\}^n$ do not hit g is at most

$$(1 - \epsilon/2)^M \leq e^{-\epsilon M/2} .$$

Since $M = c_3 cmn/\epsilon$, it follows that, for an appropriate choice of c_3, the probability that all the functions g are hit is at least

$$1 - 2^{cmn} e^{-\epsilon M/2} > 3/4 \ .$$

For the second claim, observe that a set of m randomly chosen elements from $\{0,1\}^n$ is not $\epsilon/4$-discrepant for a given function with probability at most $2^{-\Omega(m\epsilon^2)}$ (this follows from the Chernoff bound). Since

$$|\{f^{\oplus a} : a \in \{0,1\}^n\}| \leq 2^n \ ,$$

we have that the probability that a randomly chosen set of $m = \frac{1}{\epsilon^2} c_2 n$ elements of $\{0,1\}^n$ is not $\epsilon/4$-discrepant for $\{f^{\oplus a} : a \in \{0,1\}^n\}$ is at most

$$2^n \cdot 2^{-\Omega(m\epsilon^2)} \leq 1/4$$

for an appropriate choice of c_2. □

The next theorem gives a method to generate (with high probability) a hitting set and a sequence of candidates for the discrepancy test by using the output of a weak random source.

Theorem 21 *For any $\gamma > 0$, there exist a polynomial p and an algorithm which for any $\epsilon > 0$, $c > 0$, $n > 0$, and $(p(cn/\epsilon), p(cn/\epsilon)^\gamma)$-source X, given in input (ϵ, c, n, X) and having oracle access to a function $f : \{0,1\}^n \to \{0,1\}$, computes, in time polynomial in n/ϵ, sets $H, S_1, \ldots, S_k \subseteq \{0,1\}^n$ such that the following holds with probability at least $1 - 2^{-\mathrm{poly}(n)}$:*

1. *$|S_1| = |S_2| = \cdots = |S_k|$.*
2. *H is $\epsilon/2$-hitting for the set of functions g such that $K(g|f) \leq c|S_1|n$;*
3. *for some $j \in \{1, \ldots, k\}$, S_j is $\epsilon/4$-discrepant for the set of functions $\{f^{\oplus a} : a \in \{0,1\}^n\}$.*

We will use Theorem 21 by taking c as the constant c_1 introduced in the statement of Theorem 13 (Soundness of DISC-TEST).

PROOF:[Of Theorem 21] Let us apply Corollary 19 to the set I of binary strings (u,v)'s satisfying Properties 1 and 2 in Lemma 20. Then we can use a weak random source to generate sets $S_1, \ldots, S_k$ and $H_1, \ldots, H_k$ such that, with probability at least $1 - 2^{-\mathrm{poly}(n)}$, for some j, the set S_j (respectively, H_j) has the required discrepancy (respectively, hitting) property stated in Item 2 (respectively, 1) of Lemma 20. Since the hitting property is *monotone* we also have that, with at least the same probability, $H = \bigcup_j H_j$ is $\epsilon/2$-hitting for the set of functions g with $K(g|f) \leq c|S_1|n$. □

We are now ready to prove Theorem 3.

PROOF: [Of Theorem 3] We generate a set H and sets $S_1, \ldots, S_k$ as in Theorem 21. With probability at least $1 - 2^{-\mathrm{poly}(n)}$ these sets satisfy Properties $1 - 3$ of Theorem 21. From this point we assume that this is the case. We then run DISC-TEST$(f, S_i, H, \epsilon/2)$ for $i = 1, \ldots, k$ and we return S_j where j is the smallest index such that DISC-TEST(f, S_j, H, ϵ) accepts. From Theorem 17 (Completeness of DISC-TEST) and Condition 3 of Theorem 21 we have that at least one such index exists, and from Theorem 13 (Soundness of DISC-TEST) we have that the selected set is ϵ-discrepant for f. We then output $D = \mathbf{Pr}_{x \in S}[f(x) = 1]$. □

5 A New Proof of Theorem 2 and More (by L. Fortnow)

In this section we will present a simple proof of Theorem 2 due to Lance Fortnow. We first have to introduce some new notation.

For a set S and a property Π we denote by $\exists^+ x \in S.\Pi(x)$ the statement "at least half the elements of S have property Π." A promise problem [ESY84] is a pair of disjoint sets of strings (Y, N). An algorithm A solves a promise problem (Y, N) if A accepts any element of Y and rejects any element of N. Languages can be seen as a special case of promise problems where N is the complement of Y. We denote by prRP the promise version of the class RP. That is, a promise problem (Y, N) belongs to prRP if and only if there is a polynomial time algorithm $A(\cdot,\cdot)$ and a polynomial $p(\cdot)$ such that for any x of length n

$$\begin{aligned} x \in Y &\Rightarrow \exists^+ y \in \{0,1\}^{p(n)}.A(x,y) = 1 \\ x \in N &\Rightarrow \forall y \in \{0,1\}^{p(n)}.A(x,y) = 0 \end{aligned}$$

We will use the following result of Lautemann [L83] (that is an improvement on a previous result by Sipser [S83]).

Theorem 22 (Lautemann [L83]) *If $L \in \mathsf{BPP}$ then there exists a polynomial time computable Boolean function $A(\cdot,\cdot,\cdot)$ and two polynomials $p(\cdot)$ and $q(\cdot)$ such that for any x of lenght n*

$$x \in L \Rightarrow \exists^+ y \in \{0,1\}^{p(n)}.\forall z \in \{0,1\}^{q(n)}.A(x,y,z) = 1$$

$$x \notin L \Rightarrow \forall y \in \{0,1\}^{p(n)}.\exists^+ z \in \{0,1\}^{q(n)}.A(x,y,z) = 0$$

It has been observed by Lance Fortnow that Theorem 22 implies that $\mathsf{BPP} \subseteq \mathsf{RP}^{\mathsf{prRP}}$, where we denote by $\mathsf{RP}^{\mathsf{prRP}}$ the class of languages that are decidable by RP oracle machines having access to a prRP oracle.

Theorem 23 (Fortnow) $\mathsf{BPP} \subseteq \mathsf{RP}^{\mathsf{prRP}}$.

Proof: Let L be a BPP language, and let A, p and q be as in Theorem 22. Consider the following promise problem (Y, N):

$$Y = \{(x,y) : |y| = p(|x|) \wedge \exists^+ z \in \{0,1\}^{q(n)}.[A(x,y,z) = 0]\}$$

$$N = \{(x,y) : |y| = p(|x|) \wedge \forall z \in \{0,1\}^{q(n)}.A(x,y,z) = 1\}$$

By definition $(Y, N) \in \mathsf{prRP}$. In Figure 3 an RP oracle algorithm is described that solves L by using one query to (Y, N).

We now prove the correctness of the algorithm. If $x \in L$, then for at least half the choices of y we have that $(x, y) \in N$, thus the algorithm accepts with probability at least half. If $x \notin L$, then for any y we have $(x, y) \in Y$, so the algorithm accepts with probability 0. $\Box$

Theorem 2 follows from Theorem 23, since it is easy to see that if a δ-HSG exists for some constant $0 < \delta < 1$ then any RP problem and any prRP promise problem is solvable in P.

```
input : x;
begin
   Pick a random y ∈ {0,1}^{p(|x|)};
   Ask the oracle query (x, y);
   if the oracle answers YES then reject
   else accept;
end.
```

Figure 3: The $\mathsf{RP}^{\mathsf{prRP}}$ algorithm solving a generic BPP problem.

6 Conclusions

We have demonstrated how to simulate BPP algorithms in polynomial time using weak random sources of r bits and min-entropy r^γ for any $\gamma > 0$.

The main novelty in our result has been the use of *dispersers* in a context where *extractors* seemed to be necessary. Extractors have other applications besides the use of weak random sources (see, e.g., [Nis96]). It could be the case that techniques similar to ours can give stronger results or simplified proofs in these other applications as well. It remains an open question whether it is possible, for any $\gamma > 0$ and any m, to efficiently construct a $(2^r, 2^m, d, r^\gamma, 1/7)$-extractor with r and d polynomial in m. Such a construction would provide an alternative proof of the main result of this paper and would have other interesting applications.

We also emphasize that our simulation runs in NC. This is due to the parallel nature of our construction and to the fact that it is possible to give an NC construction of the SSZ-dispersers [SSZ97]. Thus, our method provides also an efficient simulation of BPNC algorithms using weak random sources.

Likewise, the proof of Lemma 1 as appeared in a preliminary version of this paper [ACRT97], as well as the proof of Theorem 2 described in Section 5, implies an NC simulation of randomized algorithms when both the algorithm and the hitting set are given as oracles. In contrast, the proof of Lemma 1 appeared in [ACR97b] seems to be inherently sequential. Andreev et al. [ACR97b] have recently used the NC proof of Theorem 2 in order to provide sufficient conditions (in terms of worst-case circuit complexity) for NC = BPNC.

Our main result (Theorem 3) can be generalized to the case where the function f that we want to sample is not Boolean but takes real values in the range $[0, 1]$. The proof of Theorem 3 contained in this paper can be easily generalized to the case of such functions. We choose however to state and prove only the case of Boolean functions since proofs are cleaner and since, as proved in [G97], sampling real-valued functions is reducible to sampling Boolean functions. We can thus get the following result as a corollary of Theorem 3 and of [G97, Theorem 5.5].

Corollary 24 *For any $\gamma > 0$, there exist a polynomial p and a deterministic algorithm A such that the following holds. For any $\epsilon > 0$, $n > 0$, any $(p(n/\epsilon), p(n/\epsilon)^\gamma)$-source X, and any $f : \{0,1\}^n \to [0,1]$, on input (ϵ, n, X) and oracle access to f, A computes, in time polynomial in n/ϵ, a value $\tilde{f}$ such that with probability at least $1 - 2^{-\mathrm{poly}(n)}$ over the outcomes of the source,*

$$|\bar{f} - \tilde{f}| \leq \epsilon$$

where $\bar{f} = 2^{-n} \sum_x f(x)$ is the average of f.

Acknowledgments

We are grateful to Oded Goldreich for several valuable comments and suggestions on preliminary versions of this paper. In particular, the use of a counting argument *á la* Kolmogorov is due to Oded. We thank Lance Fortnow for his permission to mention his results in this paper. We thank Madhu Sudan and Avi Wigderson for helpful discussions on Fortnow's result, and Michael Saks, Aravind Srinivasan and Shiyu Zhou for showing us that OR-dispersers can be obtained by an NC construction, and for other helpful conversations.

References

[ACR97a] A.E. Andreev, A.E.F. Clementi, and J.D.P. Rolim. A new general de-randomization method. *Journal of the ACM*, to appear, 1997. Preliminary version in *Proc. of ICALP'96*.

[ACR97b] A.E. Andreev, A.E.F. Clementi, and J.D.P. Rolim. Worst-case hardness suffices for derandomization: A new method for hardness vs randomness trade-offs. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, pages 177–187. LNCS 1256, Springer-Verlag, 1997.

[ACRT97] A.E. Andreev, A.E.F. Clementi, J.D.P. Rolim, and L. Trevisan. Weak random sources, hitting sets, and BPP simulations. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 264-272, 1997.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307-318, 1993.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal of Computing*, 13(4):850-864, 1984. Preliminary version in *Proc. of FOCS'82*.

[BR94] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 276–287, 1994.

[CG88] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, April 1988.

[CW89] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 14–19, 1989.

[ESY84] S. Even, A. Selman, and Y. Yacoby. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 2:159–173, 1984.

[G97] O. Goldreich. A sample of samplers — A computational perspective on sampling. *Electronic Colloquium on Computational Complexity* TR97-020, 1997.

[IW97] R. Impagliazzo and A. Wigderson. P= BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 220-229, 1997.

[L83] C. Lautemann. BPP and the Polynomial Hierarchy. *Information Processing Letters*, 17:215-217, 1983.

[LV90] M. Li and P. Vitany. Kolmogorov complexity and its applications. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A*, pages 187–254. Elsevier, 1990.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[Nis90] N. Nisan. *Using Hard Problems to Create Pseudorandom Generators.* MIT Press, 1990. ACM Distinguished Dissertations.

[Nis96] N. Nisan. Extracting randomness: How and why. In *Proceedings of the 11th IEEE Conference on Computational Complexity*, pages 44–58, 1996.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994. Preliminary version in *Proc. of FOCS'88.*

[NZ96] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996. Preliminary version in *Proc. of STOC'93.*

[S83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[SSZ95] M. Saks, A. Srinivasan, and S. Zhou. Explicit dispersers with polylog degree. In *Proceedings of the 27th ACM Symposium on Theory of Computing*, pages 479–488, 1995.

[SSZ97] M. Saks, A. Srinivasan, and S. Zhou. Personal communication, March 1997.

[SV86] M. Santha and U. Vazirani. Generating quasi-random sequences from slightly random sources. *Journal of Computer and System Sciences*, 33:75–87, 1986.

[SZ94] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 264–275, 1994.

[TS96] A. Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 276–285, 1996.

[Vaz86] U. Vazirani. *Randomness, Adversaries and Computation.* PhD thesis, University of California, Berkeley, 1986.

[Vaz87] U. Vazirani. Efficiency considerations in using semi-random sources. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 160–168, 1987.

[VV85] U. Vazirani and V. Vazirani. Random polynomial time is equal to slightly random polynomial time. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 417–428, 1985.

[Y82] A.C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 80-91, 1982.

[Zuc90] D. Zuckerman. General weak random sources. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 534–543, 1990.

[Zuc96a] D. Zuckerman. Randomness-optimal sampling, extractors and constructive leader election. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 286–295, 1996.

[Zuc96b] D. Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4/5):367–391, 1996. Preliminary version in *Proc. of FOCS'91*.