Random Walks, Conditional Hitting Sets and Partial Derandomization*

Dimitris A. Fotakis[1,2] Paul G. Spirakis[1,2]

[1] Computer Engineering and Informatics Department
University of Patras, 265 00 Rion, Patras, Greece

[2] Computer Technology Institute — CTI
Kolokotroni 3, 262 21 Patras, Greece

Email: `fotakis@cti.gr, spirakis@cti.gr`

Fax: `+30-61-993973`

Abstract. In this work we use random walks on expanders in order to relax the properties of hitting sets required for partially derandomizing one-side error algorithms. Building on a well-known probability amplification technique [AKS87, CW89, IZ89], we use random walks on expander graphs of subexponential (in the random bit complexity) size so as to avoid particular sets of "misleading" strings, while reducing the random bit complexity of the algorithm.

Then, we introduce the idea of conditional hitting sets in order to avoid the remaining "misleading" strings. In particular, given a set C, a $C_1 \subseteq C$, and an $s \notin C_1$, we suggest to exploit s for constructing a set that avoids C. Furthermore, we show how to combine random walks on expanders of subexponential size with conditional hitting sets so as to reduce the random bit complexity of any one-side error randomized algorithm.

On the other hand, an application to $\mathcal{PCP}$ systems for $\mathcal{NP}$ suggests that, if our techniques can substantially reduce the random bit complexity of arbitrary probabilistic systems, then $\mathcal{NP}$ has subexponential deterministic simulations.

Keywords: Derandomization, Hitting Sets, Random Walks

* This work was partially supported by ESPRIT LTR Project no. 20244—ALCOM–IT.

1 Introduction

A central research question in complexity theory is whether randomized algorithms, which are allowed to make desicions based on the output of a random source, are more powerful (in terms of language recognition) than deterministic algorithms of similar running time. As a consequence, many general methods have been proposed for decreasing (or even removing) the amount of random bits used by randomized algorithms.

In this direction, the idea of pseudo-random generators has been extensively used. A pseudo-random generator, starting from a short truly random seed, produces long strings that appear indistinguishable from truly random strings to any reasonably fast algorithm. The papers of [Yao82] and [BM84] introduced the idea of constucting a pseudo-random generator from cryptographically secure one-way permutations. Since then, many researchers have proposed methods for derandomizing probabilistic algorithms based on "hardness" assumptions. [Imp97] provides an excellent survey on the use of hard problems for derandomization. In this framework, hitting sets are a useful tool for obtaining derandomization results about particular complexity classes [ACR96, ACRT97].

In addition to the design of general derandomization techniques, it is important to identify the properties of the probabilistic systems that are susceptible to certain derandomization techniques. Following this direction of research, quantitative aspects of randomness in Arthur-Merlin games are studied in [BGG93]. Also, [FK95] proved that there exist natural interactive proof systems for which repetition using truly random bits can reduce the error rate to be polynomially small, while repetition using bits produced by any pseudo-random generator cannot reduce the error below a constant.

In this work, we exploit random walks on expander graphs in order to relax the properties of hitting sets that suffice for derandomization. Thus, we obtain general sufficient conditions for reducing the random bit complexity of any one-side error randomized algorithm.

1.1 Previous Work

The techniques for analyzing the "hitting" properties of random walks on expander graphs were first introduced by [AKS87]. Then, random walks on expanders used for decreasing the error rate of randomized algorithms [CW89, IZ89, MR95] and interactive proof systems [BGG93], while not substantially increasing the random bit complexity. Up to our knowledge, the first time that recursive random walks on expanders were used for decreasing the random bit complexity of certain hypothetical one-side error probabilistic systems was in [FS96].

Hitting sets are one of the basic combinatorial objects studied in the framework of derandomization [ACRT97]. Given a set S and a $S' \subseteq S$, a set H is hitting for S', if $S' \cap H \neq \emptyset$. Alternatively, H is hitting for S', if it avoids the complement of S', denoted $\overline{S'} = S - S'$, at least once. If we consider an one-side error randomized algorithm A, a set is hitting for A, if it is guaranteed to contain at least one element causing A to produce the correct outcome. Polynomial time Hitting Set Generators (HSGs) are known to derandomize one-side bounded error, polynomial time, randomized algorithms ($\mathcal{RP}$), and recently were shown to derandomize the class of two-side bounded error, polynomial time, randomized algorithms ($\mathcal{BPP}$) [ACR96].

A deterministic construction of a hitting set for combinatorial rectangles of dimension d and volume at least ϵ is presented in [LLSZ93]. For positive integers d and m, a combinatorial rectangle R within $S = [m]^d$ is defined as $R = R_1 \times \cdots \times R_d$, where, for all $i \in [d]$, $R_i \subseteq [m]$. The volume of R is $\prod_{i \in [d]} |R_i|/m^d$.

In [SSZ95], explicitly constructed dispersers were used for a polynomial simulation of any $\mathcal{RP}$ algorithm from a weak random source of min-entropy r^γ. A random source has min-entropy r^γ, if it outputs bit strings of length r and no string has probability of being output larger than 2^{r^γ} [CG88]. Disperser graphs may be thought as the analogue of hitting sets, when simulations of randomized algorithms from weak random sources are considered instead of derandomization. Additionally, a polynomial time simulation of any $\mathcal{BPP}$ algorithm from the output of a weak random source of min-entropy r^γ was obtained using some properties of hitting sets [ACRT97]. In both [SSZ95, ACRT97], the simulation asks the random source for a polynomial number of "weakly" random bits.

1.2 Summary of Results

We start with proving that a combination of the probability amplification techniques of [CW89, IZ89, BGG93] with random walks on expander graphs of subexponential size can be used for reducing the random bit complexity of one-side error algorithms fulfilling a particular condition. In fact, given a set S of strings of length r (i.e. $|S| = 2^r$), we show how to avoid any subset of S, which can be expressed as a combinatorial rectangle of dimension α, using only $\frac{r}{\alpha} + \mathcal{O}(\alpha)$ random bits. Consequently, we obtain a non-trivial upper bound on the number of random bits, that suffice for constructing hitting sets for complements of combinatorial rectangles. Additionally, we provide a simple structural description of the algorithms, whose random bit complexity can be reduced using this technique.

Given any one-side error algorithm A and any set C of strings causing A to produce wrong outcome ("misleading" strings), there always exists a $C_1 \subseteq C$, such that C_1 can be written as a combinatorial rectangle of an appropriate dimension. Therefore, random walks on expanders can be used for avoiding C_1, while reducing the random bit complexity of A. Furthermore, obtaining a string $s \notin C$ from an s_1 known not to belong to C_1 cannot be harder than obtaining an $s \notin C$ from scratch. This observation leads to the idea of conditional hitting sets, which are used for obtaining general sufficient conditions for reducing the random bit complexity of one-side error randomized algorithms. Additionally, we suggest an implementation of a conditional hitting set generator from an one-to-one function that maps the strings of C to the vertices of an expander graph G, so as the vertices of $C_2 = C - C_1$ to belong to "small" connected components of the subgraph induced by C.

Finally, an application to Probabilistic Proof Systems for $\mathcal{NP}$ implies that either the efficiency (in terms of derandomization) or the complexity (or both) of any algorithm for the construction of conditional hitting sets depend on the family of the "misleading" sets that should be avoided. Notice that similar results can easily be obtained for hitting sets.

We are not aware of any non-trivial construction of hitting sets for complements of combinatorial rectanges. Furthermore, up to our knowledge, this is the first time that random walks on expanders are combined with a generalization of hitting sets for reducing the random bit complexity of one-side error probabilistic systems. The results of [SSZ95, ACRT97] are different in nature because, if min-entropy is considered as a measure for the (truly) random bit complexity of a random string, the simulations increase the random bit complexity by a polynomial factor.

All our results exploit random walks on expander graphs of subexponential size to avoid combinatorial rectangles. Additionally, the idea of using a string not belonging to a particular subset C_1 in order to avoid the whole set C (conditional hitting set) appears for the first time in the framework of derandomization.

2 Preliminaries

In addition to the input x, a randomized algorithm A makes a series of random choices, which can be grouped together as the random string r. The output of the algorithm solely depends on the input x and the random string r. $A(x, r)$ denotes the outcome of A on input x, where the string r determines the (random) choices of A. Wlog. we only consider algorithms that produce Boolean output. An one-side bounded error algorithm A for a language L may err with probability at most $\frac{1}{4}$ in either accepting some $x \in L$, or rejecting some $x \notin L$, but not in both. In the sequel, the probability is with respect to the uniform distribution on all the binary strings of length $|r|$. [MR95] provides an excellent treatment of randomized algorithms.

A language $L \subseteq \Sigma^\star$ is in the class $\mathcal{RP}$, if there exists a worst-case polynomial time, randomized algorithm A, that for all inputs $x \in \Sigma^\star$, A always rejects all $x \notin L$, while it may reject an $x \in L$ with probability at most $\frac{1}{4}$. Similarly, the class co-$\mathcal{RP}$ consists of languages that have polynomial time, randomized algorithms erring only for $x \notin L$. The choice of the bound of $\frac{1}{4}$ on the error probability is arbitrary, because the error probability of an $\mathcal{RP}$ algorithm A can be made arbitrarily (but constantly) small by repeating A a (suitable) constant number of times.

Although the complexity classes $\mathcal{RP}$ and co-$\mathcal{RP}$ are defined in terms of decision problems, we will also use the complexity class labels for referring to algorithms. In particular, an $\mathcal{RP}$ algorithm A belongs to the class $\mathcal{RP}_\delta(r(n))$, if, for all inputs x of length n, A uses at most $\hat{r}(n) = \mathcal{O}(r(n))$ random bits, and achieves an error rate at most δ, i.e. if $x \in L$, Prob$[A(x) = \text{REJECT}] \leq \delta$.

Given an one-side error randomized algorithm A using $\hat{r}(n)$ random bits, let S be the set of all the bit strings of length $\hat{r}(n)$, and, for any fixed integer $\alpha \geq 1$, S^α be the set of all the bit strings of length $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil$. The cardinality of S is sometimes denoted by $N = 2^{\hat{r}(n)}$. A set S' is the complement of a combinatorial rectangle, if, for some integer $\alpha > 1$ and constant $\epsilon > 0$, there exists a combinatorial rectangle R within $S = \left[\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil\right]^\alpha$ of dimension α and volume ϵ, such that $S - S' \subseteq R$. Therefore, for S' being the complement of a combinatorial rectangle R, if a set H avoids R then H hits S'.

Let $0 < \epsilon \leq \frac{1}{4}$ be the error rate of A. Given an input x, there exists a set of strings $C \subseteq S$, $|C| \leq \epsilon|S|$, that cause $A(x)$ to result in wrong outcome, i.e. REJECT instead of ACCEPT if $A \in \mathcal{RP}$. We also refer to such sets C as the set of "misleading" strings for A on input x. We can define a family of sets

$$\mathcal{C} = \{C \text{ is a set of "misleading" strings for } A(x) : x \in \Sigma^\star\}$$

Obviously, $\mathcal{C}$ depends on the algorithm A, and is not necessarily equal to the family of all possible subsets of S of cardinality at most $\epsilon|S|$.

In the following, we sometime restrict our attention to $\mathcal{RP}$ and co-$\mathcal{RP}$ algorithms, because of the importance of these classes in randomized computation. However, our results hold for any one-side error randomized algorithm, independently of its worst case running time.

2.1 Probabilistically Checkable Proofs

All recent results on hardness of approximating some $\mathcal{NP}$-hard optimization problems rely on a probabilistic definition of $\mathcal{NP}$ based upon the complexity class of Probabilistically Checkable Proofs, $\mathcal{PCP}$. According to this definition, the class $\mathcal{NP}$ contains exactly those languages whose membership proofs can be checked by a polynomial time, probabilistic verifier using logarithmic number of random bits and inspecting constant number of proof bits, $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ [ALMSS92, Aro94, BGS96, Sud96].

A *verifier* V is a probabilistic polynomial time Turing machine with access to an input x, a random string r, and a proof π via an oracle. The outcome of V is either ACCEPT or REJECT. A verifier is $(r(n), q(n))$-restricted if, for all inputs x of length n, it uses at most $\hat{r}(n) = \mathcal{O}(r(n))$ random bits, and queries at most $\hat{q}(n) = \mathcal{O}(q(n))$ bits from the proof.

Definition 1 [AS92]. A language L is in $\mathcal{PCP}(r(n), q(n))$ iff there exists an $(r(n), q(n))$-restricted verifier V such that:

(a) For all $x \in L$, there exists a proof π_x that satisfies Prob$_r[V(x, r, \pi_x) = \text{ACCEPT}] = 1$,

(b) while for all $x \notin L$, every proof π satisfies Prob$_r[V(x, r, \pi) = \text{ACCEPT}] \leq \frac{1}{4}$.

Notice that a verifier V is similar to a co-$\mathcal{RP}$ algorithm, in the sense that, there exists a proof, so as V to always accept $x \in L$, while, for all proofs, V may accept some $x \notin L$ with probability at most $\frac{1}{4}$. The following lemma relates the random and the query bit complexity of a $\mathcal{PCP}$ system for a language L, with the deterministic time complexity of deciding whether an input x is in L.

Lemma 2 [ALMSS92]. *Let L be any language over Σ^* and V be a $(r(n), q(n))$-restricted verifier for L. Then for any $x \in \Sigma^*$ there exists a Boolean formula B_x on $\mathcal{O}\left(2^{\hat{r}(n)}\hat{q}(n)\right)$ variables that is satisfiable iff $x \in L$.*

2.2 The Hitting Properties of Random Walks on Expander Graphs

A technique presented in [AKS87] uses random walks on expander graphs for producing long pseudo-random bit strings.

Lemma 3 [AKS87]. *Let $\mathcal{G}$ be an infinite family of d-regular graphs with the following property: If $G = (V, E)$ is a member of $\mathcal{G}$ and A denotes its adjacency matrix multiplied by 1/d then all but the largest eigenvalue of A are less than 1 and positive.*
Then for every subset C of V with $|C| \leq |V|/16$ there exists a constant c such that the probability that a random walk on G of length $\kappa \cdot c$ arrives in every c-th step in a vertex of C is at most $2^{-\kappa}$.

The existence of families of graphs $\mathcal{G}$ satisfying the requirements of Lemma 3 is based on the existence of *constant degree expanders*. An explicit construction of constant degree expanders is given by Gabber and Galil [GG81]. The so-called Gabber-Galil expander has the advantage that we do not need to explicitly construct the entire graph. In particular, for any vertex in the expander, it is possible to compute the neighboring vertices in time polynomial in $\log |V|$. Random walks on expanders are widely used (e.g. [CW89, IZ89, BGG93, MR95]) for obtaining probability amplification results. We also use the following version of Lemma 3 that is proved in [BGG93] using the techniques of [AKS87]:

Lemma 4 [BGG93]. *For any family $\mathcal{G}$ of d-regular expander graphs there is a constant $\eta \geq 1$ such that the following is true. Suppose $\epsilon < 1/2$ and let $c = \eta \log \epsilon^{-1}$. Let $v \in \mathbb{N}$ and $C_1, \ldots, C_v$ be subsets of V having density at most ϵ. Let $b \leq v$ be an integer and $1 \leq j_1 < \ldots < j_b \leq v$ be a sequence of indices between 1 and v. Consider a random walk of length $c \cdot v$ on the expander and denote by Y_j the vertex visited at time $c \cdot j$, $j = 1, \ldots, v$. Then,* $\text{Prob}\left[Y_{j_1} \in C_{j_1}, \ldots, Y_{j_b} \in C_{j_b}\right] \leq (2\epsilon)^{b/2}$.

2.3 Hitting Sets and Derandomization

A set $H \subseteq S$ is called hitting for a $S' \subseteq S$, if $H \cap S' \neq \emptyset$. Alternatively, H is said to avoid the complement of S', denoted $\overline{S'} = S - S'$. Additionally, a set $H \subseteq S$ is called ϵ*-hitting* for security b, if for any Boolean circuit A of size at most b, $\text{Prob}_{x \in S}[A(x) = 1] \geq \epsilon$ implies that there exists at least one string $h \in H$ such that $A(h) = 1$ (e.g. [Imp97, ACRT97]). A polynomial time algorithm $\mathcal{H}$ that, given in input a number n in unary, returns a set $\mathcal{H}(n) \subseteq \{0,1\}^n$, which is ϵ-hitting for security n, is called a *quick ϵ-Hitting Set Generator* (HSG) [ACRT97].

If we restrict our attention to $\mathcal{RP}$ algorithms, given an $A \in \mathcal{RP}$, a set $H \subseteq S$ is called ϵ-hitting for A, if, for any input x, $\text{Prob}_{r \in S}[A(x, r) = \text{ACCEPT}] \geq \epsilon$ implies that there exists at least one $h \in H$ such that $A(x, h) = \text{ACCEPT}$. If C is a set of "misleading" strings for the algorithm A, a hitting set H for A is required to hit the complement of C, i.e. $\overline{C} = S - C$. H is sometimes said to avoid the set of "misleading" strings C. Similar definitions can also be obtained for co-$\mathcal{RP}$ algorithms, and in general, for any one-side error randomized algorithm. Obviously, if quick HSGs exist for the whole class $\mathcal{RP}$, then $\mathcal{P} = \mathcal{RP}$. Moreover, quick HSGs are shown to derandomize $\mathcal{BPP}$ [ACR96], and some properties of hitting sets are used for obtaining polynomial time simulations of

$\mathcal{BPP}$ using weak random sources [ACRT97]. On the other hand, quick HSGs are unlikely to exist for arbitrary probabilistic systems, even if the random bit complexity is logarithmic. Otherwise, an application of such HSGs to $(\log n, 1)$-restricted verifiers would imply some unexpected bounds on the deterministic time complexity of the whole $\mathcal{NP}$ (e.g. $\mathcal{NP} \subseteq \mathcal{P}$).

In this paper, we introduce the idea of *conditional hitting sets* in order to partially derandomize (i.e. reduce the random bit complexity) one side-error randomized algorithms. At first, we show that, given any one-side error algorithm A, there exists a subset C_1 of any set C of "misleading" strings for A, such that C_1 can be avoided with probability at least $1 - \epsilon$ using only a constant fraction of the random bits used by A. Since a set avoiding $C_2 = C - C_1$ is not guaranteed to also avoid C_1, we suggest to exploit a string $s \notin C_1$ in order to construct a hitting set for $\overline{C} = S - C$. Loosely speaking, this seems more difficult than avoiding C_2, but easier than avoiding the whole C, because a piece of information about C_1 is available, since C_1 has already been avoided by s.

Definition 5. Given any one-side error randomized algorithm A, any set of "misleading" strings (for A) C, a subset $C_1 \subseteq C$, and a string $s \in S$, a set $W_{C_1,s}$ is said to be a conditional hitting set for A, if $s \notin C_1$ implies that $W_{C_1,s}$ is a hitting set for A.

The dependence on the algorithm A may be implicit. In particular, we sometimes refer to a $W_{C_1,s}$ as a hitting set for $\overline{C} = S - C$, where C is any set of "misleading" strings of some algorithm A. Obviously, a hitting set for A is also a conditional hitting set for A corresponding to the partition $C_1 = \emptyset, C_2 = C$. Therefore, conditional hitting sets generalize the idea of hitting sets, and any quick HSG is also a quick Conditional Hitting Set Generator (CHSG). However, we do not know whether, for all $C_1 \subseteq C$ and randomized algorithms A, the existence of a quick CHSG corresponding to C_1 would imply the existence of a quick HSG.

3 Using Random Walks for Reducing Randomness

The following probability amplification result is a consequence of Lemma 3. We provide a simple proof for the sake of completeness.

Lemma 6. *Let A be an $\mathcal{RP}_{1/16}(r(n))$ algorithm for a language L. Then for* any $0 < \delta < \frac{1}{16}$, *there exists an $\mathcal{RP}_\delta$ algorithm A' for L that uses $\hat{r}(n) + c \log d \left(\log \frac{1}{\delta}\right)$ random bits (c, d are the constants of Lemma 3).*

Proof. The algorithm A' with the desired properties will result from multiple invocations of A. In particular, A' proceeds as follows:

(1) It constructs a d-regular expander graph $G(V, E)$ that contains $2^{\hat{r}(n)}$ vertices and fulfills the hypothesis of Lemma 3.

(2) It reads a random string r of length $\hat{r}(n) + c \log d \left(\log \frac{1}{\delta}\right)$, c is the constant of Lemma 3.

(3) It performs a random walk on the graph G. Let $\kappa \cdot c$ be the length of the random walk. Let r_i be the bit string that is associated with the vertex reached by the random walk at the $(i \cdot c)$-th step, $i = 0, 1, \ldots, \kappa$. (r_0 corresponds to the initial vertex determined by $\hat{r}(n)$ truly random bits). Then A is invoked with the same input x for each bit string r_i ($\kappa + 1$ times). A' rejects x iff A rejects x for all (pseudo) random strings r_i.

Since error probability of A is at most $\frac{1}{16}$, there exists a $C \subseteq V, |C| \leq |V|/16$ such that a vertex $u \in C$ iff it is associated with a bit string r_i causing $A(x, r_i)$ to result in wrong outcome (i.e. REJECT instead of ACCEPT). Let κ be $\log \frac{1}{\delta}$. Then Lemma 3 implies that the probability of all bit strings $r_i \in C$ is $P_{\text{error}} \le 2^{-\log(1/\delta)} = 2^{\log \delta} = \delta$. Furthermore, a (truly random) bit string of length $\hat{r}(n) + c \log d \left(\log \frac{1}{\delta}\right)$ completely determines a walk of length $c \log \frac{1}{\delta}$ on G.

If the error rate of the algorithm A is $\frac{1}{16} < \epsilon < \frac{1}{2}$, we can use Lemma 4 instead of Lemma 3 so as to obtain an algorithm A' that uses at most $\hat{r}(n) + 2\eta \log d \left(\log \frac{1}{\epsilon}\right) \left(\log_{1/2\epsilon} \frac{1}{\delta}\right)$ random bits, and achieve error probability at most δ. □

Next, we show that random walks on expander graphs can be exploited for decreasing the number of truly random bits used by any $\mathcal{RP}$ algorithm A which satisfy a particular condition. Moreover, this can be performed without increasing the error probability of A. Let A be any $\mathcal{RP}(r(n))$ algorithm and, given an input x, $C \subset S$ be all the random strings that cause $A(x)$ to produce wrong outcome (i.e. REJECT instead of ACCEPT). In the sequel, we only consider $\mathcal{RP}$ algorithms A such that, ginen any input x, the corresponding set C of "misleading" strings is characterized by the following:

Assumption 7. *There exist an integer $\alpha > 1$ and sets $C_1^\alpha, C_2^\alpha, \ldots, C_\alpha^\alpha \subseteq S^\alpha$, such that a bit string $s \in C$ iff s can be written as $s_1 \circ s_2 \cdots \circ s_\alpha$, for some $s_1 \in C_1^\alpha, s_2 \in C_2^\alpha, \ldots, s_\alpha \in C_\alpha^\alpha$.*

Obviously, Assumption 7 restricts the set of "misleading" random strings to the bit strings produced by the concatenation of substrings $s_i \in C_i^\alpha$, $C_i^\alpha \subseteq S^\alpha$, $i = 1, \ldots, \alpha$. Consequently, C is contained in a combinatorial rectangle of dimension α within S that should be avoided at least once, i.e. the complement of C, $\overline{C} = S - C$, should be hit. Then, we prove that Assumption 7 is a sufficient condition for reducing the number of random bits used by A.

Theorem 8. *Let A be an $\mathcal{RP}(r(n))$ algorithm for a language L. If A fulfills Assumption 7 for some integer $\alpha > 1$, then there exists an $\mathcal{RP}$ algorithm $\hat{A}$ for L that uses at most $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil + \Theta(\gamma\alpha)$ random bits, and achieves error rate $2^{-\gamma}$, for any integer $\gamma \ge 1$. Moreover, $\hat{A}$ invokes A at most $\Theta(\gamma\alpha)$ times.*

Proof. At first, we obtain an algorihm A' for L using at most $r_{A'} = \hat{r}(n) + 4(\alpha+1)c \log d$ random bits and achieving error rate at most $2^{-4(\alpha+1)}$ (Lemma 6). Moreover, the proof of Lemma 6 implies that A' invokes A at most $4(\alpha+1)$ times. Wlog. we assume that $4(\alpha+1)c\log d \le \left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil$.

We prove that Assumption 7 allows a (pseudo) random string for a run of A' to be produced by a random walk on an expander graph containing $2^{\lceil \hat{r}(n)/\alpha \rceil}$ vertices. In particular, the algorithm $\hat{A}$ operates on any input x as follows:

(1) It constructs a $\hat{d}$-regular graph $\hat{G}(\hat{V}, \hat{E})$ that fulfills the hypothesis of Lemma 4. The graph $\hat{G}$ contains a vertex for every possible bit string of length $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil$. Let $\hat{c}$ be the constant of Lemma 4.

(2) $\hat{A}$ performs a random walk on the graph $\hat{G}$. Let $\gamma(\alpha+1)\hat{c}$ be the length of the random walk, and s_i be the bit string associated with the vertex reached by the $(i \cdot \hat{c})$-th step of the walk, $i = 1, \ldots, \gamma(\alpha+1)$. Bit strings r_j, $|r_j| \ge r_{A'}$, $j = 0, \ldots, \gamma - 1$, are constructed by the concatenation of $\alpha + 1$ consecutive results of the random walk, $r_j = s_{j(\alpha+1)+1} \circ s_{j(\alpha+1)+2} \cdots \circ s_{j(\alpha+1)+\alpha+1}$.

(3) $\hat{A}$ invokes $A'(x)$ γ times with (pseudo-)random strings r_j, $j = 0, \ldots, \gamma - 1$, and rejects x iff all invocations of A' rejects x.

Let C be the set of the "misleading" strings for $A'(x)$. Since Assumption 7 holds for A and $(\alpha+1)\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil \ge r_{A'}$, there should exist sets $C_1^\alpha, C_2^\alpha, \ldots, C_\alpha^\alpha, C_{\alpha+1}^\alpha \subseteq S^\alpha$, such that a bit string $s \in C$ iff $s = s_1 \circ s_2 \cdots \circ s_\alpha \circ s_{\alpha+1}$, for some $s_1 \in C_1^\alpha, s_2 \in C_2^\alpha, \ldots, s_{\alpha+1} \in C_{\alpha+1}^\alpha$. This implies that

A' results in wrong outcome iff all the strings $s_{j(\alpha+1)+i}$ produced by the random walk of the step (2) belong to the sets C_i^α, $j = 0, \ldots, \gamma - 1$, $i = 1, \ldots, \alpha + 1$.

Furthermore, $|C_1^\alpha||C_2^\alpha| \cdots |C_{\alpha+1}^\alpha| \leq 2^{-4(\alpha+1)}|\hat{V}|^{\alpha+1}$, beacuse the error rate of A' is at most $2^{-4(\alpha+1)}$. Hence, for at least one $1 \leq i \leq \alpha+1$, the cardinality of C_i^α should be at most $|\hat{V}|/16$. An application of Lemma 4 implies that all γ invocations of A' err with probability at most $2^{-\gamma}$.

Obviously, the algorithm A is invoked exactly $4(\alpha+1)\gamma$ times, and $\hat{A}$ uses at most $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil + \gamma(\alpha+1)\hat{c}\log\hat{d}$ random bits. □

Clearly, the proof of Theorem 8 also applies to any one-side error randomized algorithm, including algorithms in co-$\mathcal{RP}$, i.e. polynomial time algorithms always accepting inputs $x \in L$, but also accepting with probability at most $\frac{1}{4}$ inputs $x \notin L$.

The previous theorem implies that Assumption 7 describes a family of sufficient conditions for reducing the need of true randomness in one-side error randomized algorithms. However, even for small constant values of α, Assumption 7 is quite general and restrictive. Hence, it is not expected to hold for arbitrary values of the efficiency parameter α. On the other hand, any polynomial time randomized algorithm can be reduced to a Boolean circuit $B(Y_1, Y_2, \ldots, Y_\lambda)$, $\lambda = \mathcal{O}(\text{poly}(n))$. In general, each Y_i is a random variable denoting the outcome of a probabilistic test invoked by the algorithm. Therefore, the actual values of Y_i's depend on the input x, the random string r, and the particular probabilistic test. It is not hard to verify the following proposition:

Proposition 9. *Assumption 7 holds with parameter α for any one-side error randomized algorithm $B(Y_1, Y_2, \ldots, Y_\lambda)$, such that*

(a) Y_i's can be partitioned into independent groups of random variables, and

(b) the outcome of each group only depends on at most $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil$ positions of the random string.

Many natural one-side error, randomized algorithms fulfill Proposition 9 and Assumption 7 for small integers $\alpha > 1$. Therefore, Theorem 8 can be applied to such algorithms in order to reduce the need of true randomness.

4 General Conditions for Reducing Randomness

Given an algorithm A and an integer $\alpha > 1$, for any set of "misleading" strings C, there exists a $C_1 \subseteq C$, such that C_1 consists of all the strings $s \in C$ that fulfill Assumption 7 with the given parameter α. Then, Theorem 8 shows how a randomized algorithm $\hat{A}$, that uses less random bits than A, can avoid C_1 with probability at least $1 - 2^{-\gamma}$. Moreover, assume that the strings of $C_2 = C - C_1$ are distributed in the vertices of the expander graph G (proof of Lemma 6) such that, in the subgraph G_C induced by the vertices of C, the vertices of C_2 are only contained by "small" connected components. Provided that C_1 is avoided, one can exhaustively invoke A on all the vertices belonging to a "small" connected component of G_C so as to avoid the whole C.

Then, we exploit the aforementioned idea so as to derive a family of general conditions for reducing randomness in one-side error randomized algorithms. These conditions are general in the sense that they can be applied to any one-side error algorithm, although their efficiency still depends on the algorithm (see also Theorem 14 and Corollary 15). We further generalize these techniques by introducing the notion of conditional hitting sets.

Let A be any one-side error algorithm using $\hat{r}(n)$ random bits and achieving error rate at most ϵ, $0 < \epsilon \leq \frac{1}{4}$, and $G(V, E)$ be a d-regular expander graph on $N = 2^{\hat{r}(n)}$ vertices. Also, let $F : S \mapsto V$ be an one-to-one, computable function, that maps the bit strings of length $\hat{r}(n)$ to the vertices of G, and for any set of "misleading" strings $C \subset S$, $|C| \leq \epsilon N$, let $G_i(V_i, E_i)$, $F(C) = \bigcup_i V_i$, be the connected components of the induced subgraph $G_{F,C}(F(C), E_{F(C)})$. Given an integer pair (α, g),

$1 < \alpha,\ g < N$, consider a partition of C into sets C_1, C_2, so as, for some sets $C_1^\alpha, C_2^\alpha, \ldots, C_\alpha^\alpha \subseteq S^\alpha$, $C_1 = C \cap C_1^\alpha \times C_2^\alpha \times \cdots \times C_\alpha^\alpha$, and $C_2 = C - C_1$. Alternatively, there exist sets $C_1^\alpha, C_2^\alpha, \ldots, C_\alpha^\alpha \subseteq S^\alpha$, such that a string $s \in C_1$ iff $s \in C$ and $s = s_1 \circ s_2 \cdots \circ s_\alpha$, for some $s_1 \in C_1^\alpha, s_2 \in C_2^\alpha, \ldots, s_\alpha \in C_\alpha^\alpha$. Suppose that the bit strings of S can be distributed (by F) to the vertices of the d-regular expander graph G in order the following to be true:

Assumption 10. *Given an one-side error algorithm A, and an integer pair (α, g), $1 < \alpha,\ g < N$, for any set of "misleading" strings $C \subset S$, $|C| \leq \epsilon N$, there exist a partition of C into C_1, C_2 as above, and an one-to-one computable function $F : S \mapsto V$, such that for all $s \in C$ at least one of the following holds:*

(1) $s \in C_1$, i.e. $s = s_1 \circ s_2 \cdots \circ s_\alpha$, for some $s_1 \in C_1^\alpha, s_2 \in C_2^\alpha, \ldots, s_\alpha \in C_\alpha^\alpha$.

(2) $F(s)$ belongs to a connected component $G_i(V_i, E_i)$ of the subgraph induced by $F(C)$ with $|V_i| \leq g$ ("small" connected component).

Notice that Assumption 10 is a generalization of Assumption 7, because the former with parameters $(\alpha, 1)$, $C_2 = \emptyset$, and any function F, implies the latter with parameter α. Therefore, the following theorem is a generalization of Theorem 8.

Theorem 11. *Let A be any one-side error randomized algorithm for a language L, that runs in time $T_A(n)$, uses $\hat{r}(n)$ random bits and achieves an error rate ϵ, $0 < \epsilon \leq \frac{1}{4}$. Also, for $N = 2^{\hat{r}(n)}$, an integer pair (α, g), $1 < \alpha,\ g < N$, and a d-regular expander graph $G(V, E)$, $|V| = N$, let $F : S \mapsto V$ be a function that fulfills Assumption 10 with parameters (α, g) and can be constructed in time $T_F(N)$.*
Then, there exists a one-side error randomized algorithm $\tilde{A}$ for L, that runs in time $\mathcal{O}(\gamma \alpha g T_A(n) + T_F(N) + \mathrm{poly}(N))$, uses at most $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil + \Theta(\gamma\alpha)$ random bits, and achieves error rate $2^{-\gamma}$, for any integer $\gamma \geq 1$.

Proof. In the following, we assume that $A \in \mathcal{RP}$. However, the same arguments hold for any one-side error, randomized algorithm. The algorithm $\tilde{A}$ will be the result of multiple invocations of A. In particular, given an input x, $|x| = n$, and a random string r of length $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil + \Theta(\gamma\alpha)$, for some integer $\gamma \geq 1$, the algorithm $\tilde{A}$ proceeds as follows:

(1) It constructs a $\hat{d}$-regular graph $\hat{G}(\hat{V}, \hat{E})$ that fulfills the hypothesis of Lemma 4. The graph $\hat{G}$ contains a vertex for every possible bit string of length $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil$. Let $\hat{c}$ be the constant of Lemma 4.

(2) It constructs a d-regular graph $G(V, E)$, $|V| = N = 2^{\hat{r}(n)}$, that fulfills the hypothesis of Lemma 4 (and Lemma 3). Let c be the constant of Lemma 4. Also, the algorithm $\tilde{A}$ constructs an one-to-one computable function $F : S \mapsto V$ that fulfills Assumption 10 with parameters (α, g). Obviously, both steps (1) and (2) can be performed in time $\mathcal{O}(T_F(N) + \mathrm{poly}(N))$. In the sequel, wlog. we assume that an access to F can be done in time $\mathcal{O}(1)$.

(3) $\tilde{A}$ performs a random walk of length $\gamma(\alpha + 1)\hat{c}$ on the graph $\hat{G}$. Let s_i be the bit string associated with the vertex reached by the $(i \cdot \hat{c})$-th step of the walk, $i = 1, \ldots, \gamma(\alpha + 1)$, and r_j, be the bit strings constructed by the concatenation of $\alpha + 1$ consecutive s_i's, $r_j = s_{j(\alpha+1)+1} \circ s_{j(\alpha+1)+2} \cdots \circ s_{j(\alpha+1)+\alpha+1}$, $j = 0, \ldots, \gamma - 1$.

(4) $\tilde{A}$ uses pseudo-random strings r_j for performing a random walk of length $\alpha \cdot c$ on the graph G. Let r_i^j be the outcome of the the $(i \cdot c)$-th step, and $W_1 = \{r_i^j \in S : j = 0, \ldots, \gamma - 1 \text{ and } i = 0, \ldots, \alpha\}$. $A(x)$ is invoked for each bit string of W_1 as random string. If A accepts x for some string of W_1, then $\tilde{A}$ accepts x and terminates.

(5) For each $v_l = F(s_l)$, $s_l \in W_1$, the algorithm $\tilde{A}$ computes a tree (V_l, E_l) of G that contains exactly $g+1$ vertices including v_l. Let $W_2 = \{s \in S : F(s) \in V_l - \{v_l\}\}$. $A(x)$ is invoked for each bit string of W_2 as random string. $\tilde{A}$ rejects x iff all invocations of A for strings in W_2 reject x. The steps (3), (4) and (5) can be performed in time $\mathcal{O}(\gamma\alpha g T_A(n) + \mathrm{poly}(n))$.

Clearly, the time complexity of $\tilde{A}$ is $\mathcal{O}(\gamma\alpha g T_A(n) + T_F(N) + \mathrm{poly}(N))$. Moreover, $\tilde{A}$ uses at most $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil + \gamma(\alpha+1)\hat{c}\log\hat{d}$ random bits for the random walk on $\hat{G}$.

Let C be the set of the "misleading" strings of A with input x, and C_1, C_2 be any partition of C for which F fulfills Assumption 10 with parameters (α, g). The proof of Theorem 8 implies that all the strings $s \in W_1$ belong to the set C_1 with probability at most $2^{-\gamma}$. Furthermore, if at least one string $s_l \in W_1$ belongs to $S - C_1$, then either $s_l \notin C$ or $s_l \in C_2$. In the latter case, Assumption 10 implies that any string that belongs to C but not to C_1, it is contained by a "small" ($|V_l| \le g$) connected component in the subgraph G_C induced by C. Hence, there exists at least one string $s'_l \in V_l - \{s_l\} \subseteq W_2$ that does not belong to C, and the error probability of $\tilde{A}$ is at most $2^{-\gamma}$. □

Notice that the explicit construction of the function F requires at least exponential time on the number of random bits used by A. Therefore, for general $\mathcal{RP}$ (co-$\mathcal{RP}$) algorithms A, a $\mathrm{poly}(\hat{r}(n))$ time algorithm for the implicit construction of F is required so as $\tilde{A}$ to be an $\mathcal{RP}$ (co-$\mathcal{RP}$) algorithm.

If $2^{\hat{r}(n)/\alpha} g = 2^{\hat{r}(n)}$, then the random bit complexity of A can be reduced to $\hat{r}(n)/\alpha$ by invoking A exactly g times for all the possible values of the remaining $\hat{r}(n)\left(1 - \frac{1}{\alpha}\right)$ positions. Theorem 11 may be thought as a non-trivial generalization of this crude derandomization technique.

Theorem 11 suggests the use of random walks on expanders of subexponential size in combination with a conditional hitting set for the complement of $C = C_1 \cup C_2$, so as to reduce the randomness used by the algorithm A. Theorem 8 shows that short random seeds can be used by random walks on expander graphs in order to avoid the "misleading" strings that fulfill Assumption 7. Hence, we can construct a set W_1, that avoids C_1 with probability at least $1 - 2^{-\gamma}$, while reducing the random bit complexity of the algorithm A. Moreover, Assumption 10 can be used for constructing a set of size g that avoids $C_2 = C - C_1$. However, if C_1 is not empty, given an arbitrary set W'_2 that avoids C_2, it may not be the case that $W_1 \cup W'_2$ avoids the whole set C. Theorem 11 suggests to exploit the strings $s \notin C_1$ (than can be found using only a fraction of the random bits used by A) in order to construct a set that avoids the whole C, i.e. a hitting set for $\overline{C} = S - C$.

The proof of Theorem 11 implies that Assumption 10 can be used for the construction of a conditional hitting set of size g for $\overline{C}$. In particular, for each $s \notin C_1$, the corresponding component V_s of W_2 is a hitting set for $\overline{C}$. Therefore, $W_1 \cup W_2$ is a hitting set for $\overline{C}$ with probability $1 - 2^{-\gamma}$. In order to prove that a tree on $g+1$ vertices of G is a conditional hitting set for $\overline{C}$, we exploit a function F that distributes the strings of S to the expander graph G according to Assumption 10.

Obviously, given any one-side error algorithm, the techniques above can also be applied for any construction of conditional hitting sets from strings $s \notin C_1$. Therefore, Theorem 11 holds for any one-side error, randomized algorithm that achieve error rate ϵ and fulfill the following generalization of 10.

Assumption 12. *Given an algorithm A and integers (α, g), $1 < \alpha$, $g < |S|$, for any set of "misleading" strings $C \subset S$, $|C| \le \epsilon|S|$, there exist $C_1^\alpha, \ldots, C_\alpha^\alpha \subseteq S^a$ defining a partition of C into $C_1 = C \cap C_1^\alpha \times \cdots \times C_\alpha^\alpha$ and $C_2 = C - C_1$, such that, for all $s \in C$, either $s \in C_1$, or a (conditional) hitting set W, $|W| \le g$, for $\overline{C} = S - C$ can be computed from s in time $T_W(|S|)$.*

Corollary 13. *Let A be any one-side error randomized algorithm for a language L, that runs in time $T_A(n)$, uses $\hat{r}(n)$ random bits, achieves an error rate ϵ, $0 < \epsilon \le \frac{1}{4}$, and fulfills Assumption 12 for some integers (α, g).*
Then, there exists an one-side error randomized algorithm $\tilde{A}$ for L, that runs in time $\mathcal{O}\Big(\gamma\alpha g T_A(n) + \gamma\alpha T_W(2^{\hat{r}(n)}) + \mathrm{poly}(n)\Big)$, uses at most $\left\lceil \frac{\hat{r}(n)}{\alpha} \right\rceil + \Theta(\gamma\alpha)$ random bits, and achieves error rate $2^{-\gamma}$, for any integer $\gamma \ge 1$.

Proof. The proof is similar to the proof of Theorem 11, apart from the steps (2) and (5). In the step (2), we do not further need to explicitly construct the expander graph G and compute the function F. Thus, we save an additive factor of $\mathcal{O}(\mathrm{poly}(N) + T_F(N))$ in the time complexity of the algorithm $\tilde{A}$. In the step (5), the set W_2 consists of the union of the conditional hitting sets constructed from each $s \in W_1$. Since $|W_1| = \mathcal{O}(\gamma\alpha)$, $|W_2| = \mathcal{O}(\gamma\alpha g)$. Moreover, Assumption 12 implies that W_2 can be constructed in time $\mathcal{O}\left(\gamma\alpha T_W(2^{\hat{r}(n)})\right)$. □

4.1 The Complexity of Computing Conditional Hitting Sets

Next, we apply the techniques above to $\mathcal{PCP}$ systems for $\mathcal{NP}$ in order to show that for a wide range of parameters, the time complexity of computing conditional hitting sets for arbitrary probabilistic systems bounds from above the deterministic time complexity of the whole class $\mathcal{NP}$. Since there exist $\mathcal{PCP}$ systems for $\mathcal{NP}$ that use logarithmic randomness (cf. [ALMSS92, Aro94, BGS96, Sud96]), the sizes of the expander graphs and the hitting sets are polynomial in the input x. The following theorem suggests that Assumption 12 cannot hold for arbitrary families of sets of "misleading" strings $\mathcal{C}$, even if $|S|$ is polynomial in the input x.

Theorem 14. *If Assumption 12 holds with parameters (α, g) for arbitrary families of "misleading" sets $\mathcal{C}$ that contain elements of polynomial size, then there exist contants $\rho, \mu > 1$ such that* 3-SAT *is included in $\mathcal{DTIME}\left(2^{2^{\mathcal{O}(\alpha)} n^{\rho/\alpha} \mu g} + \alpha g\, \mathrm{poly}(n) + \alpha T_W(n^\rho)\right)$.*

Proof. For some appropriately chosen constants ρ, μ, ϵ, let V be a polynomial time restricted verifier for 3-SAT, that uses $\rho \log n$ random bits, inspects μ proof bits, and achieves error rate $0 < \epsilon \leq \frac{1}{4}$. Obviously $|S| = n^\rho$. If Assumption 12 holds with parameters (α, g) for arbitrary families $\mathcal{C}$ of "misleading" sets C, $|C| = \mathrm{poly}(n)$, then we can apply Corollary 13 to the verifier V in order to obtain a verifier $\tilde{V}$ for 3-SAT, that uses at most $\frac{\rho}{\alpha} \log n + \mathcal{O}(\alpha)$ random bits and inspects at most $\mathcal{O}(\alpha\mu g)$ proof bits. Moreover, Assumption 12 and Corollary 13 imply that the verifier $\tilde{V}$ runs in time $\mathcal{O}(\alpha g\, \mathrm{poly}(n) + \alpha T_W(n^\rho)))$. Given any 3-SAT instance x, an application of Lemma 2 to the verifier $\tilde{V}$ results in a Boolean formula B_x of $\mathcal{O}\left(2^{\mathcal{O}(\alpha)} n^{\rho/\alpha} \mu g\right)$ variables, such that B_x is satisfiable iff x is satisfiable. □

Corollary 15. *For any $g = \mathrm{o}(n)$, there exists a constant $\alpha_g^\star$ such that, for any $\alpha \geq \alpha_g^\star$, Assumption 12 does not hold with parameters (α, g) for arbitrary families $\mathcal{C}$, unless $\mathcal{NP}$ is included in $\mathcal{DTIME}\left(2^{\mathrm{o}(n)} + \mathrm{poly}(n) + T_W(n^\rho)\right)$.*

Proof. For any language L in $\mathcal{NP}$, there exist constants ρ, μ, ϵ such that L has a polynomial time restricted verifier V, that uses $\rho \log n$ random bits, inspects μ proof bits, and achieves error rate ϵ. Suppose that Assumption 12 holds with parameters $g = \mathrm{o}(n)$, and $\alpha = \kappa\rho$, for some constant $\kappa \geq 2$. Then, Corollary 12 can be applied to V in order to obtain a verifier $\tilde{V}$ for L, that uses at most $\frac{\log n}{\kappa} + \mathcal{O}(\kappa\rho)$ random bits, and inspects $\mathcal{O}(\kappa\rho\mu g)$ proof bits. Furthermore, $\tilde{V}$ runs in time $\mathcal{O}(\mathrm{poly}(n) + T_W(n^\rho))$. Hence, given an input x, we can apply Lemma 2 to the verifier $\tilde{V}$ so as to obtain a Boolean formula B_x on $\mathrm{o}(n)$ variables such that, x is in L iff B_x is satisfiable. □

5 Conclusions

In this work, we show how to use a particular technique for generating long dependent pseudorandom bit strings in order to relax the properties of hitting sets that suffice for partial derandomization. Consequently, we obtain general sufficient conditions for reducing the random bit complexity of any one-side error randomized algorithm. Our results are based on the use of random walks on expanders of subexponential size for avoiding combinatorial rectangles in combination with conditional hitting sets.

Conditional hitting sets are not equivalent to hitting sets, unless, for all algorithms A and definitions of C_1, the existence of a CHSG implies the existence of a HSG. Therefore, the following question naturally arises from the definition of conditional hitting sets: Given an one side error algorithm A and any set C of "misleading" strings for A, does there exist a $C_1 \subseteq C$ such that a set avoiding C_1 helps in constructing a set avoiding C (i.e. a hitting set for the algorithm A)? A research direction resulting from this question is whether, for any algorithm A, the existence of a quick CHSG corresponding to some particular partition of C (e.g. Assumption 10) implies the existence of a quick HSG for A. A positive result would imply that, for the corresponding definition of $C_1 \subseteq C$, the information that a string s is not in C_1 does not help in avoiding the whole C. A complementary research direction is to obtain some non-trivial definitions for C_1 so as a conditional hitting set for a randomized algorithm A to be strictly easier to be constructed than a hitting set. A definition of C_1 should be thought as non-trivial, if avoiding the whole C is (strictly) more difficult than avoiding C_1 (e.g. Assumptions 10 and 12 provide non-trivial definitions for C_1).

Acknowledgements: We would like to thank Jose Rolim for his helpful comments on an earlier draft of this paper.

References

[AKS87] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic simulation in logspace. *Proc. of the 19th ACM Symposium on Theory of Computing*, pp. 132–140, 1987.

[ACR96] A. Andreev, A. Clementi, and J. Rolim. Hitting Sets Derandomize $\mathcal{BPP}$. *Proc. of the 23rd International Colloquium on Automata, Languages and Programming*, pp. 357–368, 1996.

[ACRT97] A. Andreev, A. Clementi, J. Rolim, and L. Trevisan. Weak Random Sources, Hitting Sets, and $\mathcal{BPP}$ Simulations. *Proc. of the 38th IEEE Symposium on Foundations of Computer Science*, pp. 264–272, 1997.

[Aro94] S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems.* PhD Thesis, UC Berkeley, 1994.

[ALMSS92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Proc. of the 33th IEEE Symposium on Foundations of Computer Science*, pp. 14–23, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of $\mathcal{NP}$. *Proc. of the 33th IEEE Symposium on Foundations of Computer Science*, pp. 2–13, 1992.

[BGG93] M. Bellare, O. Goldreich, and S. Goldwasser. Randomness in Interactive Proofs. *Computational Complexity* **3**, pp. 319–354, 1993.

[BGS96] M. Bellare, O. Goldreich, and M. Sudan. Free Bits, $\mathcal{PCP}$s and Non-Approximability—Towards Tight Results (3rd Version). TR 95–024. Electronic Colloquium on Computational Complexity. December 1995.

[BM84] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, **13**, pp. 850–864, 1984.

[CG88] B. Chor and O. Goldreich. Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity. *SIAM Journal of Computing*, **17**(2), pp. 230–261, 1988.

[CW89] A. Cohen and A. Widgerson. Dispersers, Deterministic Amplification, and Weak Random Sources. *Proc. of the 30th IEEE Symposium on Foundations of Computer Science*, pp. 14–19, 1989.

[FK95] U. Feige and J. Kilian. Impossibility Results for Recycling Random Bits in Two-Prover Proof Systems. *Proc. of the 27th ACM Symposium on Theory of Computing*, pp. 457–468, 1995.

[FS96] D. Fotakis and P. Spirakis. $(\mathrm{poly}(\mathrm{loglog}\, n), \mathrm{poly}(\mathrm{loglog}\, n))$–restricted verifiers are unlikely to exist for languages in $\mathcal{NP}$. *Proc. of the 21th Mathematical Foundations of Computer Science*, pp. 360–371, 1996.

[GG81] O. Gabber and Z. Galil. Explicit constructions of linear–sized superconcentrators. *Journal of Computer and System Sciences* **22**, pp. 407–420, 1981.

[Imp97] R. Impagliazzo. Using Hard Problems to Derandomize Algorithms: An Incomplete Survey. *Proc. of the 1st Symposium on Randomization and Approximation Techniques in Computer Science*, pp. 165–173, 1997.

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

[LLSZ93] N. Linial, M. Luby, M. Sacks, and D. Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Proc. of the 25th ACM Symposium on Theory of Computing*, pp. 258–267, 1993.

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

[SSZ95] M. Sacks, A. Srinivasan, and S. Zhou. Explicit Dispersers with polylog degree. *Proc. of the 27th ACM Symposium on Theory of Computing*, pp. 479–488, 1995.

[Sud96] M. Sudan. *Efficient checking of polynomials and proofs, and the hardness of approximation problems.* ACM Distinguished Theses series, Lecture Notes in Computer Science, Vol. 1001, Springer 1996.

[Yao82] A. Yao. Theory and Applications of Trapdoor Functions. *Proc. of the 23rd IEEE Symposium on Foundations of Computer Science*, pp. 80–91, 1982.