Deterministic Simulation of Probabilistic Constant Depth Circuits (Preliminary Version)

Miklos Ajtai

Avi Wigderson

IBM Research - San Jose

ABSTRACT

We explicitly construct, for every integer n and $\epsilon > 0$, a family of functions (psuedo-random bit generators) $f_{n,\epsilon}:\{0,1\}^{n^\epsilon} \rightarrow \{0,1\}^n$ with the following property: for a random seed, the pseudo-random output "looks random" to any polynomial size, constant depth, unbounded fan-in circuit. Moreover, the functions $f_{n,\epsilon}$ themselves can be computed by uniform polynomial size, constant depth circuits.

Some (interrelated) consequences of this result are given below.

1) *Deterministic simulation of probabilistic algorithms.* The constant depth analogues of the probabilistic complexity classes RP and BPP are contained in the deterministic complexity classes $DSPACE(n^\epsilon)$ and $DTIME(2^{n^\epsilon})$ for any $\epsilon > 0$.

2) *Making probabilistic constructions deterministic.* Some probablistic constructions of structures that elude explicit constructions can be simulated in the above complexity classes.

3) *Approximate counting.* The number of satisfying assignments to a (*CNF* or *DNF*) formula, if not too small, can be arbitrarily approximated in $DSPACE(n^\epsilon)$ and $DTIME(2^{n^\epsilon})$, for any $\epsilon > 0$.

We also present two results for the special case of depth 2 circuits. They deal, respectively, with finding a satisfying assignment and approximately counting the number of assignments. For example, for 3-CNF formulas with a fixed fraction of satisfying assignmemts, both tasks can be performed in polynomial time!

1. INTRODUCTION

The relationship between randomized and deterministic computation is a fundamental issue in the theory of computation. The results on this subject fall into the following categories.

Simulating randomness by nonuniformity

Adleman [*Ad*] showed that any language in RP can be computed by a polynomial size family of circuits. However, the proof is existential, and there is no known way of explicitly constructing these circuits. A similar result, for simulating probabilistic, polynomial size, constant depth circuits by nonuniform deterministic ones is due Ajtai and Ben-Or [*AB*].

0272-5428/85/0000/0011$01.00 © 1985 IEEE

Simulating randomness under an unproven assumption

Yao [*Ya*] has shown that if one-way functions exist, then RP is contained in $DTIME(2^{n^\epsilon})$, for any fixed positive ϵ. Note that the assumption is extremely strong, as it implies in particular that $P \neq NP \cap coNP$. Similar results are given in [*FLS*], who study the space complexity of the simulation, and [*RT*], who consider RNC instead of RP.

Simulating randomness by alternation

Sipser and Gacs [*Si*] showed that BPP is contained in Δ_2^P. Of course, the time or space complexities of languages in this class are unknown. A related result, due to Stockmeyer [*St*], is that approximate counting is in Δ_3^P.

Simulating specific randomized algorithms

By a careful analysis of how randomness is used in a specific algorithm, one may be able to replace it by a deterministic construction. Such examples are the parallel algorithms in [*Lu, KUW, KW*]. Also related are explicit constructions of graphs with special properties, which can be found in [*Ma*] and [*GG*].

There were no explicit upper bounds on the deterministic simulation of any nontrivial class of probabilistic algorithms. In fact, there is no such simulation that does less than brute force enumeration of all possibilities for the random inputs.

We prove in this paper that probabilistic, polynomial size, constant depth, unbounded fan-in circuits can be simulated in $DSPACE(n^\epsilon)$, (and hence in $DTIME(2^{n^\epsilon})$), for every fixed positive ϵ. This is done by generating a small set of pseudo-random binary strings, such that a randomly chosen one of them "looks random" to any polynomial size, constant depth circuit.

It is interesting to note that our "pseudo-random bit generator" is purely combinatorial, in contrast to the number theoretic generators used in cryptography (e.g. [*Sh, BM, BBS*]).

The proof that our generator "works" requires an intimate understanding of the structure of constant depth circuits. Such an understanding is drawn from the lower bound proof techniques for such circuits [*Aj, FSS*]. Moreover, these lower bounds are all "probabilistic" (or "non constructive"), and an essential part of building the generator is making them explicit. To this end we use the idea of "k-wise independent" random variables (e.g. see [*ACGS, Lu, An, KUW*]).

In section 2 we give definitions and state our main theorem. In section 3 we discuss applications of the main theorem, and in section 4 we sketch its proof. In section 5 we obtain refined results on depth-2 circuits, and discuss their applications.

2. DEFINITIONS AND THE MAIN THEOREM

A *circuit* C is a directed acyclic graph with node labels. The nodes of indegree zero are labeled with input variables, the nodes of outdegree zero are labeled with output variables, and the rest of the nodes are labeled from $\{AND, OR, NOT\}$. We put no bound on fan-in or fan-out.

The *size* of a circuit C, $s(C)$, is the number of nodes in it. The *depth* of C, $d(C)$, is the length of the longest input-output path. We say that C is an $(s,d)-circuit$ if $s(C) \leq s$ and $d(C) \leq d$.

We shall be interested in families of circuits. Let $s,d:N \to N$ be functions. We say that $\{C_n\}$, $n = 1,2,...$ is an $(s,d)-family$ if for all n, $s(C_n) \leq s(n), d(C_n) \leq d(n)$. If $s = n^{O(1)}, d = O(1)$ then $\{C_n\}$ is a *PC family* (Polynomial size, Constant depth). A family is *uniform* if there exists a Turing machine that on input n in unary, outputs a description of $\{C_n\}$, using only $O(\log n)$ space.

We shall mainly deal with one-output circuits. Every circuit C with n inputs computes a function $C:\{0,1\}^n \to \{0,1\}$ in a natural way. Define $p(C) = Pr[C(x) = 1]$, where $x \in \{0,1\}^n$ with uniform probability.

For inputs that are generated pseudo-randomly, we use the following. Let $f:\{0,1\}^m \to \{0,1\}^n$ be a function. Define $p_f(C) = Pr[C(x) = 1]$, where $x = f(y)$ and $y \in \{0,1\}^m$ with uniform probability.

Two important parameters measure the "goodness" of f as a pseudo-random bit generator for a circuit C. The natural one is $|p(C) - p_f(C)|$. Another parameter, for which we get better bounds, is how small can $p(C)$ get so that still $p_f(C)$ does not vanish.

We can now state the main theorem.

Main Theorem: Let k,d,ℓ,ε be fixed. Then there exists a family of functions $\{f_n:\{0,1\}^{n^\varepsilon} \to \{0,1\}^n\}$, $n = 1,2,...$, depending only on the parameters above, with the following properties:

(*i*) $\{f_n\}$ can be computed by a uniform, PC family of circuits. (So in particular, $\{f_n\}$ can be computed in LOGSPACE).

(*ii*) Let $\{C_n\}$ be any (n^k,d) family of circuits. Then for every n,

a) $p(C_n) - p_{f_n}(C_n) \le n^{-\ell}$

b) For a fixed $\delta = \delta(k,d,\varepsilon)$, if $p(C_n) \ge 2^{-n^\delta}$, then $p_{f_n}(C_n) > 0$

3. APPLICATIONS

The applications are given not necessarily in order of importance, but rather in order of notational convenience. All the applications are based on the fact that we can get a fairly good approximation of the output behaviour of C_n by "testing" it on only 2^{n^ε} inputs.

The following notation will be used often. Let $g:N \to [0,1]$ be a function. We say that g is polynomially small if $g(n)^{-1} = n^{O(1)}$. We say that g is sub-exponentially small if $g(n)^{-1} = o(2^{n^\varepsilon})$, for every fixed $\varepsilon > 0$.

3.1. Approximate counting

Let the number of satisfying assignments to a (CNF or DNF) formula F be $\#F$. Computing $\#F$ from F is #P-complete. It is not known whether #P is in the polynomial time hierarchy. An easier problem is approximate counting, which in this case, is to find an integer in the range$[(1+\beta)^{-1}\#F, (1+\beta)\#F]$. Call this β-approximation. For any polynomially small β, β-approximation was shown to be in Δ_3^p by Stockmeyer [*S*]. No explicit deterministic upper bounds were known for approximate counting. Let $p(F) = \frac{\#F}{2^n}$ be the fraction of satisfying assignments of F, where n is the number of variables in F.

Corollary 1: Consider formulas F with polynomially small $p(F)$. Then for every fixed ε and every polynomially small β, the β-approximation problem for F is in $DSPACE(n^\varepsilon)$ (and hence also in $DTIME(2^{n^\varepsilon})$).

Proof (sketch): F is a polynomial size, depth 2 circuit. In $DSPACE(n^\varepsilon)$ all the "seeds" y of f_n can be generated, $f_n(y)$ computed and tested on F. The output is $(\sum_{|y|=n^\varepsilon} F(f_n(y)))2^{n-n^\varepsilon}$. By (*ii*), part *a*) of the main theorem, it is the desired approximation.

3.2 Easy cases of satisfyability

If we are just interested in finding a satisfying assignment to a formula F, the result above can

be improved. In [VV], Valiant and Vazirani showed that finding a satisfying assignments in formulas with exactly one such assignment is essentially as hard as the general case. The following result, which complements theirs, says that if the number of satisfying assignments is large enough, then satisfyability becomes easier.

Corollary 2: Consider formulas F with sub-exponentially small $p(F)$. Then a satisfying assignment of F can be found in $DSPACE(n^{\epsilon})$ (and $DTIME(2^{n^{\epsilon}})$).

This result follows from (*ii*), part *b*) of the main theorem. It will be refined in the section on depth-2 circuits.

3.3 Making Probabilistic Constructions Deterministic

Following Sipser [*Si*], we define a probabilistic construction to be a language $L \subset \{0,1\}^* \times \{0,1\}^*$ with the property that if $(u,v) \in L$, then $Pr[(u,x) \in L] \geq \frac{1}{2}$, where x is uniformly chosen with $|x| = |v|$. (u usually gives the size of the required object in unary, and then a random object of the right size has the desired properties). The deterministic construction problem for L is, on input u, to generate v s.t. $(u,v) \in L$.

If $L \in \Sigma_i^p$, Sipser calls it a Σ_i^p*-construction*. He shows that if L is a Σ_i^p-construction, then the deterministic construction problem for L is in Σ_{i+2}^p. (An analogous statement is true for Π_i^p).

Note that $L \in \Sigma_i^p$ or $L \in \Pi_i^p$ means that L can be recognized by a family of constant depth (but possibly exponential size) circuits. We say that L is a PC construction if it can be recognized by a uniform PC family of circuits.

Corollary 3: If L is a PC construction, then the deterministic construction problem for L is in $DSPACE(n^{\epsilon})$ (and in $DTIME(2^{n^{\epsilon}})$).

Note that the uniformity is needed for our deterministic machine to generate the circuit recognizing constructions of size $|u|$, where u is the input.

3.4 Deterministic Simulation of Probabilistic Constant Depth Circuits

A probabilistic circuit C is one with "real" input variables, z, and random input variables, x. If $|z| = n$ and $|x| = m(= m(n))$, C computes a function $C:\{0,1\}^{n+m} \to \{0,1\}$. Let $p(C(z)) = Pr[C(z;x) = 1]$ when $x \in \{0,1\}^m$ with uniform probability. The idea of recognizing languages by probabilistic circuits is that the behaviour of $p(C(z))$ will depend on whether z is in the language or not.

We define two families of complexity classes, $PC1(\alpha)$ and $PC2(\alpha)$ where PC refers to polynomial size, constant depth, 1 and 2 refer to whether we allow one- or two-sided errors, and α is the "accuracy" (in general $\alpha:N \to [0,1]$ is a function).

A language $L \subset \{0,1\}^*$ is in PC1(α) if there exists a uniform PC family $\{C_n\}$ s.t. for every n and every $z \in \{0,1\}^n$

$$z \in L \to p(C(z)) \geq \alpha(n)$$
$$z \notin L \to p(C(z)) = 0.$$

A language $L \subset \{0,1\}^*$ is in PC2(α) if there exists a uniform PC family $\{C_n\}$ s.t. for every n and every $z \in \{0,1\}^n$

$$z \in L \to p(C(z)) \geq \frac{1}{2} + \alpha(n)$$

$$z \notin L \to p(C(z)) \leq \frac{1}{2} - \alpha(n).$$

Corollary 4: For every fixed $\epsilon > 0$ we have
(*i*) for every sub-exponentially small function α, $PC1(\alpha) \subset DSPACE(n^{\epsilon})$
(*ii*) for every polynomially small function α, $PC2(\alpha) \subset DSPACE(n^{\epsilon})$

4. SKETCH OF PROOF OF THE MAIN THEOREM

The key notion in the proof is that of approximating a circuit. A given circuit will undergo a series of simplifications, each restricting the inputs, that will change the output behaviour by only tiny amounts.

The proof has two logical parts. In part I we show how to approximate $n^{1-\epsilon}$ input bits of a PC circuit by only $O(\log n)$ bits. In part II we show how to iterate this construction, adding only a constant to the depth and a polynomial to the size.

PART I

We need some notation. Let C be a PC circuit on n variables $A = \{x_1,...,x_n\}$. For any subset $Y \subset A$ and a binary vector v of length $|Y|$, let $C_{Y,v}$ denote the circuit obtained from C by assigning the values v to the variables in Y, in the natural order. Note that $C_{Y,v}$ has $|A - Y|$ variables.

Lemma 1 asserts that, for a random set of all but $n^{1-\epsilon}$ of the input variables and random assignments to them, the resulting circuit will depend only on a constant number of inputs (although it has $n^{1-\epsilon}$ of them).

Definition 1: We say that a circuit C depends on t variables if there exists a subset $T \subset A$ of size t s.t. for every assignment to the variables in T, the resulting circuit is constant. We denote the minimum such t by $t(C)$, and some T of this cardinality by $T(C)$.

Lemma 1: Let d,l,u be positive integers and $\epsilon > 0$. Then there exists an integer t so that if n is sufficiently large; C is a (n^l,d)-circuit with n input variables A, and $W \subset A$ is random with $|W| = n^{1-\epsilon}$, then $Pr[t(C_{A-W,v}) > t] \leq n^{-u}$, where v is a random, uniformly chosen assignment to $A - W$. Moreover, with probability at least $1 - n^{-u}$ the set W will satisfy $Pr[t(C_{A-W,v}) \leq t] \geq 1 - 2^{-n^{\frac{\epsilon}{t^2}}}$. If W satisfies this property, call it t-local.

The proof of lemma 1 is an inductive argument on the depth of the circuit, similar in flavour to the lower bound proofs in [*Aj, FSS, Yao*]. In fact, the first part of lemma 1 appears with a different proof in [*Aj*]. The inductive step is based on a property of depth-2 circuits, which is given in theorem 1 in section 5.

Note that different choices of v may result in different subsets $T(C_{A-W,v})$ of W that the resulting circuit depends on. However, lemma 1 tells us that the output distribution of C will essentially remain unchanged if instead of assigning random values to the variables in W (not $A - W$), we use assignments that "look random" only on t-subsets of W. This motivates the next definitions and corollary.

Definition 2: A random variable $Z = Z_1,Z_2,...,Z_m$ with $Z_i \in \{0,1\}$ is said to be (m,t,p)-uniform if for all $1 \leq i \leq m$ $Pr[Z_i = 1] = p$ and for every t-subset I of $[m]$, the variables $\{Z_i \mid i \in I\}$ are mutually independent. When $p = .5$ we say that Z is (m,t)-uniform.

The key fact about (m,t)-uniform sequences is that they can be simply generated from only $t \log m$ random bits by PC circuits, using polynomials over finite fields. (e.g. see [*KUW*]).

Definition 3: Let $W \subset A$ and integer t be fixed. A random variable $Y \in \{0,1\}^n$ is called (W,t)-local if $Y|_W$ is $(|W|,t)$-uniform, $Y|_{A-W}$ is uniform, and these two restricted random variables are independent.

Corollary 5: Let d,l,u,ϵ,t be as in lemma 1. For each possible W of size $n^{1-\epsilon}$ let Y_W be (W,t)-local. Then (1) if W is t-local, then $|p(C) = - Pr[C(Y_W) = 1]| \leq 2^{-n^{\frac{\epsilon}{2}}}$. (2) If W is

chosen uniformly at random, then we have $|p(C) - P_r[C(Y_W) = 1]| \leq 1 - n^{-u}$.

We are now in a situation where, if given a t-local set W, we can replace its $n^{1-\epsilon}$ random bits by $O(\log n)$ random bits. Corollary 1 shows that most W will work, but to use that we need as many as $(n^{1-\epsilon}\log n)$ bits. Our next step will be to generate t-local sets W pseudo-randomly, using only $O(\log n)$ random bits. This is done by extracting from the proof of lemma 1 only the essential properties of the random variable W that are actually used.

Definition 4: Let X be a random variable whose values are subsets of a set A of size n. We say that X has the small intersection property with parameters α,β,t if for every set $V \subset A$ with $|V| \leq n^{\beta}$ we have that if $s \leq t$, then $P_r[|V \cap X| \leq s] \geq 1 - n^{-(1-\alpha-\beta)s}$.

Definition 5: The random variable $X \subset A$ has the d-iterated small intersection property with parameters α,β,t if there exists a sequence of random variables $X_1,X_2,\ldots,X_d$ so that $X = X_d$, $X_{i+1} \subset X_i$, and for any possible fixed values $B_1,\ldots,B_i$ of the variables $X_1,\ldots,X_i$ we have that X_{i+1} with the conditions $X_1 = B_1,\ldots,X_i = B_i$ has the small intersection propetry with parameters α,β,t.

Lemma 2: The consequences of lemma 1 and corollary 5 hold if we replace a randomly chosen W of size $n^{1-\epsilon}$ by a random variable W that has the d-iterated small intersection property with parameters $1 - \epsilon, \frac{\epsilon}{2}, t$.

Note that in definition 4, only intersections of cardinality t or less are important. From this it is easy to deduce:

Lemma 3: If W is an $(n,n^{-\epsilon},t)$-uniform random variable, then W has the small intersection property with parameters $1 - \epsilon, \frac{\epsilon}{2}, t$.

Again, such a random variable can be constructed from $t\log n$ random bits by PC circuits. To get a random variable with the d-iterated small intersection property one can use d independent constructions as above, which require only $dt\log n$ random bits.

To summarize the first part of the proof, we have shown how to replace $n^{1-\epsilon}$ random bits by $O(\log n)$ random bits, thus reducing the number of inputs by roughly $n^{1-\epsilon}$.

PART II

At this point it is natural to iterate the construction roughly n^{ϵ} times. However, this presents some difficulties. For example, if we implement the construction in the first part, the depth of the circuit increases by a constant, and so we cannot repeat that more than a constant number of times. Another problem is that bounds the number of iterations is that we must keep the circuit polynomial in the number of remaining inputs, so we have to stop when at least a polynomial fraction remains.

Conceptually performing this iterative process, we obtain a sequence of roughly n^{ϵ} pseudo-random subsets of variables, that together with the remaining part (of size roughly n^{ϵ}) form a partition of the set of variables. To each pseudo random subset we assign (independently) a pseudo-random assignment (requiring total of $O(n^{\epsilon}\log n)$ bits, and to the remaining subset assign random values (only n^{ϵ}).

In order to perform this process in constant depth, we shall generate all parts in the partition together with their assignments simultanously. We first define the partition-assignment pair abstractly, as random variables, and then show how they can be generated from n^{ϵ} random bits.

Definition 6: Let d,u,t be integers and $\delta > 0$. For every n and $0 \leq \mu \leq n$ define $<F,P>$ to be a

fooling pair of random variables if the following conditions hold:

(1) each value of F is 0,1 assignment to the variables in A, $|A| = n$.

(2) each value $P = <P_0,...,P_\mu>$ is a sequence of subsets of A so that $P_0,...,P_\mu$ form a partititon of A.

(3) for all $0 \le i < \mu$ if $A_0,...,A_{i-1}$ are fixed subsets of A then the random variable P_i with the condition $P_0 = A_0,...,P_{i-1} = A_{i-1}$ has the d-iterated small intersection property with parameters $1 - 2\delta,\delta,t$.

(4) for all $0 \le i \le \mu$ if $A_0,...,A_i$ are fixed subsets of A then with the condition $P_0 = A_0,...,P_i = A_i$ the random variables $F|_{A_0},...,F|_{A_i}$ are independent.

(5) for all $0 \le i < \mu$ if $A_i \subset A$ then $F|A_i$ with condition $P_i = A_i$ is an $(|A_i|,t)$-uniform random variable.

(6) for each A_μ s.t. that the random variable P_μ can take, $F|_{A_\mu}$ with the condition $P_\mu = A_\mu$ may have a uniform distribution over over all assignments to A_μ or not. The probability that it does is at least $1 - n^{-u-2}$.

(7) $|P_\mu| \ge n^\delta$

The technical properties of the fooling pair guarantee that it fools any PC circuit with the appropriate parameters.

Lemma 4: For all d,l,u,δ there exists a t such that for all sufficiently large $n,\mu \le n$ and a fooling pair $<F,P>$ we have the following. For every (n^l,d)-circuit C with n inputs, $|p(C) - Pr[C(F) = 1]| \le n^{-u}$.

The proof of this lemma will be by induction, that will show that the simultaneous construction/definition of the fooling partition-assignment pair actually simulates the natural iterative construction. For this we need the following definition and lemma.

Definition 7: For all $0 \le i \le \mu$ let Y_i be the assignment to the variables in A which coincides to F on $\cup\{P_i \mid j < i\}$ and takes random values uniformly and independently of $<F,P>$ on $A - \cup\{P_i \mid j < i\}$.

Lemma 5: For all but a fraction n^{-u-2} of the values B that P may take, and for all $0 \le i \le \mu - 1$ we have $|Pr[C(Y_i) = 1] - Pr[C(Y_{i+1}) = 1]| \le n^{-u-2}$, when these probabilities are conditioned by the event $P = B$.

Note that, conditioned on the event $P = B$, we have $Pr[C(Y_0) = 1] = p(C)$ and $Pr[C(Y_{\mu-1}) = 1] = Pr(C(F) = 1)$ so Lemma 5 implies Lemma 4.

Proof of Lemma 5: In the following proof all probabilities are considered with the condition $P = B$. Let $F_i = F|_{\cup\{P_j \mid j<i\}}$. Then for every value B that Pmay take $Pr(C(Y_i) = 1) = \sum_f Pr(F_i = f)Pr(C(Y_i) = 1 \mid F_i = f)$ where f takes all of the possible values for F_i.

Suppose now that f is fixed. We may consider C as a circuit with the variables $A - \cup\{P_i \mid j < i\}$ only, if we evaluate the remaining variables according to f. Since P_i has the d-iterated small intersection property (even if F_i is fixed) then lemma 2 implies that for all but a fraction n^{-u-3} of values B that P may take, given the event $F_i = f$ we have $|Pr[C(Y_i) = 1] - Pr[C(Y_{i+1}) = 1]| \le n^{-u-3}$

Lemma 6 deals with the explicit construction of a fooling pair from a small number of random bits.

Lemma 6: Let d,t,u be integers and $0 \le \delta \le .01$. Then for every large enough n, a fooling pair $<F_n,P_n>$ with $\mu = n^{10\delta}$ can be constructed by a

LOGSPACE uniform PC family of circuits, given as inputs $O(\mu \log n) + n^{50\delta}$ random bits. (The implicit constant depends only on the fixed parameters above.)

The proof of lemma 6 is a messy exercise in logic design.

The main theorem, except property (*ii*)b now follows by choosing $\delta = .01\varepsilon$ and lemmas 4 and 6. The proof of this property follows the same outline, only using the stronger assertion in lemma 1.

5. DEPTH-2 CIRCUITS

The two results in this section are algorithms for the problems of approximate counting and finding a satisfying assignment, respectively, in depth-2 circuits (CNF/DNF formulae). The two important parameters which affect the running time of the algorithms are the fraction of satisfying assignments and the sizes of clauses.

Recall that $p(F) = Pr[F(x) = 1] = \frac{\#F}{2^n}$, if n is the number of variables in F. A CNF formula F is k-CNF if every clause is of size at most k. Similarly we define a k-DNF formula.

Theorem 1 deals with the approximation of depth 2 circuits. It shows that the output almost always depends on a small subset of the input variables.

Theorem 1: Let F be a k-DNF formula on inputs A, and m any number sufficiently bigger than k. ($2^m > (km2^{k+1})^k$ will do) Then there exists a subset $S \subset A$, s.t. $|S| \leq m^{k^2}$ with the following property. If v is a random, uniformly chosen binary string of size $|S|$ then $Pr[F_{S,v}$ *is non constant*$] \leq 2^{-m}$. Moreover, such a subset S can be found in $DTIME(n^{k^2})$.

This theorem can be used as an algorithm for approximate counting (see section 3.1).

Corollary 6: Let F be any k-CNF or k-DNF formula, and assume wlog that $p = p(F) \leq .5$. Then the β-approximation problem for F can be solved in $DTIME(n^{k^2} + 2^{(\log \frac{\beta}{p})^{k^2}})$.

For example, for 3-CNF formulas with a fixed fraction of assignments, the β-approximation problem can be solved in polynomial time for every fixed $\beta \geq 0$.

Theorem 2 gives an algorithm for finding a satisfying assignment.

Theorem 2: Let F be a k-CNF formula on n variables with $p = p(F)$. Then we can find a satisfying assignment of F in $DTIME(n^{k2^k(\log p^{-1})})$. ($p$ is unknown to the algorithm !)

For example, this theorem says that 3-CNF instances of SAT with a fixed fraction of satisfying assignments are easy, as we can find one in polynomial time!

REFERENCES

[*Ad*] L. Adleman, "Two theorems on random polynomial time", 19th FOCS (1978), 75-83.

[*Aj*] M. Ajtai, "Σ_1^1-formulae on finite structures", Annals of Pure and Applied Logic 24 (1983), 1-48.

[*An*] R. Anderson, "Set splitting", manuscript.

[*AB*] M. Ajtai and M. Ben-Or, "A theorem on probabilistic constant depth computations", 16th STOC (1984), 471-474.

[*ACGS*] W. Alexi, B. Chor, O. Goldreich and C. P. Schnorr, "RSA/Rabin bits are $\frac{1}{2} + \frac{1}{poly(\log n)}$ secure, 25th FOCS (1984), 449-457.

[*BBS*] L. Blum, M. Blum and M. Shub, "A simple, secure pseudo-random number generator", Advances in Cryptography, Proc. of CRYPTO-82 (1982), 61-78.

[*BM*] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits", Proc of the 23rd FOCS, (1982) 112-117.

[*ES*] P. Erdos and P. Spencer, "Probabilistic methods in Combinatorics", Academic Press, (1974).

[*FLS*] M. Furst, R. J. Lipton and L. Stockmeyer, "Pseudo-random number generation and space complexity", Information and Control, to appear.

[*FSS*] M. Furst, J. B. Saxe and M. Sipser, "Parity, Circuits, and the polynomial time hierarchy", 22nd FOCS (1981), 260-270.

[*Gi*] J. Gill, "Complexity of probabilistic Turing machines", SIAM J. of Computing 6 (1977), 675-695.

[*GG*] O. Gaber and Z. Galil, "Explicit construction of linear sized superconcentrators", J. Comp. Sys Sci. 22 (1981), 407-420.

[*KUW*] R. M. Karp, E. Upfal and A. Wigderson, "The complexity of parallel computation on matroids", submitted to the 26th FOCS.

[*KW*] R. M. Karp and A. Wigderson, "A fast parallel algorithm for the maximal independent set problem", 24th STOC, (1984), 266-272.

[*Lu*] M. Luby, "A simple parallel algorithm for the maximal independent set problem", 17th STOC (1985).

[*Ma*] G.A. Margulis, "Explicit constructions of graphs without short cycles and low density codes", Combinatorica 2, 1 (1982), 71-78.

[*Ru*] W. L. Ruzzo, "On uniform circuit complexity", J. Comput. Sys. Sci., 22, 3, (1981), 365-383.

[*RT*] J. H. Reif and J. D. Tygar, "Towards a theory of parallel randomized computation", TR-07-84, Aiken Computation Lab., Harvard University (1984).

[*Si*] M. Sipser, "A complexity theoretic approach to randomness", 15th STOC (1983), 330-335.

[*Sh*] A. Shamir, "On the generation of cryptographically strong pseudo-random sequences, 8th ICALP, Lecture notes in Comp. Sci., 62, Springer-Verlag (1981), 544-550.

[*St*] L. Stockmeyer, "The complexity of approximate counting", 15th STOC (1983), 118-126.

[*VV*] L. G. Valiant and V. V. Vazirani, "NP is as easy as detecting unique solutions", 17th STOC (1985).

[*Ya*] A. C. Yao, "Theory and applications of trapdoor functions", Proc. of the 23rd FOCS, (1982), 80-91.