Random Polynomial Time is Equal to Slightly-random Polynomial Time

Umesh V. Vazirani *
University of California, Berkeley

Vijay V. Vazirani **
Cornell University

Abstract

Random Polynomial Time (R_p) is currently considered to be the class of *tractable* computational problems. Here one assumes a source of truly random bits. However, the known sources of randomness are imperfect. They can be modeled as an adversary source, called slightly-random source. Slightly-random Polynomial Time (SR_p) is the class of problems solvable in polynomial time using such a source. SR_p is thus a more realistic definition of a tractable computational problem. In this paper we give an affirmative answer to the question "*is* $R_p = SR_p$?" Our proof method is constructive: given an R_p algorithm for a problem, we show how to obtain an SR_p algorithm for it.

Studying the relationship between randomized and deterministic computation is currently an important issue. A central question here is "*is* $R_p = P$?" Our result may be a step towards answering this question.

G Supported by NSF-Grant MCS 82-04506, and by the IBM Doctoral Fellowship.

(2*) Supported by NSF Grant BCR 8503611, and an IBM Young Faculty Fellowship.

1. Introduction

An important goal of theoretical computer science is to capture theoretically the notion of a *tractable* computational problem. It is currently well-accepted that a problem is tractable if there is a polynomial time algorithm for it, having the ability of flipping a fair coin at each step. This formalization is theoretically appealing, and captures important problems such as primality testing [Ra2, SS], and testing polynomial identities [Sc], which are not known to be efficiently solvable without randomization. † However, it is beset with a difficulty: the available sources of randomness such as Zener diodes, and Geiger counters are imperfect. They do not output unbiased, independent coin-flips. Santha and Vazirani [SV] introduce a general model for such imperfect sources of randomness: the *slightly-random source*. Determining the usefulness of such a source is of theoretical as well as practical importance In particular, Vazirani [Va] defines the class Slightly-random Polynomial Time (SR_p), the class of functions

† In the case of primality testing, an efficient algorithm is known, assuming the extended Riemann Hypothesis [Mi].

0272-5428/85/0000/0417$01.00 © 1985 IEEE

computable in polynomial time using a slightly-random source, and asks "is $R_p = SR_p$?", where R_p is the class of problems solvable in polynomial time, using a fair coin [Gi, Ra1]. We give an affirmative answer to this question. Our proof method is constructive: given an R_p algorithm for a problem, we show how to obtain an SR_p algorithm for it.

In recent years, the use of randomized arguments has played a crucial role in theoretical computer science, and the relationship between randomized and deterministic computation has become a fundamental issue. A central question here is 'is $R_p = P$?'. Important insights into this question have been provided by Adleman [Ad] who showed that problems in R_p can be solved with a family of polynomial sized circuits, and Yao [Ya] who showed that R_p is contained in $DTIME\ (2^{n^\epsilon})$, for any fixed $\epsilon > 0$, if one-way functions exist.

The class SR_p is intermediate between P and R_p. The computational time available is the same in all three classes; however, compared to a fair coin, the slightly-random source is a provably weaker source of random bits. Hence answering the "$R_p = SR_p$?" question may be a step towards answer the "$R_p = P$?" question.

Other insights on the relationship between randomized and deterministic computation have been provided in the following works: Sipser [Si] showed that BPP is contained in Δ_2^p. Using similar techniques, Stockmeyer [St] shows that approximate counting is in Δ_3^p. Valiant and Vazirani [VV] show that the problem *UNIQUE SATISFIABILITY* is complete in D^p under randomized reductions. On the other hand, Blass and Gurevich [BG] show an oracle relative to which this problem is not complete in D^p under deterministic reductions, thereby suggesting that randomized reducibilities may indeed be more powerful than deterministic ones. Karp and Wigderson [KW], and Luby [Lu] give methods of dispensing with randomness in algorithms, in special cases. Ajtai and Wigderson [AW] give deterministic simulations of constant depth probabilistic circuits.

2. The Slightly-Random Source

Imperfect sources of randomness suffer from two shortcomings: *bias* and *dependence*. If the bits output are independent, the problem of bias is easy to deal with [vN]. However, the problem becomes more serious if the bias of the next bit depends on the bit-sequence generated so far. Blum [Bl] considers one such situation, models the imperfect source as a finite state Markov process, and gives a simple and efficient algorithm for obtaining truly random bits from it. Santha and Vazirani [SV] deal with the problem in its full generality: the case when the bias of the next bit is a function of the entire bit-sequence generated so far.

This model, called the *slightly-random source*, may be thought of as an *adversary source*. The adversary has complete knowledge of the manner in which we will use its bit-sequence and has unlimited computing power. Moreover, the adversary knows the bit-sequence generated so far. To generate the next bit it fixes the bias of a coin, flips it, and outputs faithfully the outcome. The only restriction on the adversary is that the bias must be in the range δ and $1\text{-}\delta$, for some fixed constant δ, $0 < \delta \leq 1/2$. The function giving the bias of the next bit, given the bit-sequence generated so far, is called the *strategy* of the adversary. Thus the adversary is free to choose any strategy to foul our application requiring random bit-sequences. Since the slightly-random source models physical sources of randomness, such as Zener diodes, determining its usefulness is of theoretical as well as practical importance.

Some basic questions about slightly random sources have been answered recently:

a) Can we obtain 'better' bits from the output of a slightly random source? In particular, is there a boolean function f, f: n-bit sequences $\rightarrow$ {0,1}, that converts the adversary's n-bit sequence into a bit having bias in the smaller range $[\varepsilon, 1-\varepsilon]$, where $\delta < \varepsilon < 1-\delta$. Santha and Vazirani [SV] show that for every function f the adversary has a strategy to force ε to be at most δ.

b) What if we had two *independent* slightly random sources? Surprisingly enough, in this case 'better' bits can be obtained. Let x and y be n-bit sequences output by the two sources. Vazirani [Va] shows that the bit obtained by computing the $GF[2]$ inner product $x \cdot y$ has a smaller bias. In fact, the bias can be made arbitrarily small by choosing n suitably large – small enough so that the bit sequence obtained is *quasi-random* [SV]. Quasi-random sequences are indistinguishable for random sequences in a strong sense [SV] and thereby can be used in applications such as randomizing algorithms.

The slightly-random source is perhaps the simplest model of an adversary source of random bit-sequences; it is certainly possible to consider variants on this model. An interesting variant is considered by Chor and Goldreich [CG]. Their adversary has even more power: it is required to bound the probability of blocks of bits, rather than single bits. Our proof showing that BPP can be simulated with a slightly-random source (see Section 5) can be generalized for this adversary source also. This source has some interesting theoretical properties, but appears too strong for obtaining 'better' bit-sequences. The only known method involves a probabilistic construction.

3. Slightly-random Polynomial Time

Definition [Gi]: A language $L \subseteq \{0,1\}^*$ is in Random Polynomial Time (R_p) if:

there exists a polynomial-time algorithm T which can obtain the flip of a fair coin at each step:

(i) If $x \in L$ $Pr[T \text{ accepts } x] \geq 1/2$

(ii) If $x \notin L$ T always rejects x.

Slightly-random polynomial time is defined analogously except that the fair coin is replaced by an adversary controlled coin—the slightly-random source.

Definition [Va]: A language $L \subseteq \{0,1\}^*$ is in Slightly-random Polynomial Time (SR_p) if:

for every δ there exists a polynomial-time algorithm T, which at each step can obtain one bit from a slightly-random source having bias in the range [δ, 1-δ]:

(i) If $x \in L$ $Pr[T \text{ accepts } x] \geq 1/2$

(ii) If $x \notin L$ T always rejects x.

Clearly $P \subseteq SR_p \subseteq R_p$. Are these inclusions proper? Blum and Vazirani [BV] show that the problem of testing whether a given polynomial is identically zero is in SR_p. This problem is in R_p [Sc], but is not known to be in P. In this paper, we show how to convert any R_p algorithm into an SR_p algorithm that recogonizes the same language.

During the course of its computation on input x, the random polynomial time algorithm T can flip the coin polynomially many times, say $|x|^t = m$ times. Without loss of generality, we may assume that T begins its computation by flipping the coin m times and recording the outcomes of the flips. Thereafter, T proceeds deterministically. T is equivalent to a deterministic polynomial time algorithm M which takes two inputs x and r. M uses the second input r in place of the coin tosses to simulate T on input x. This gives the following equivalent definition of R_p:

Definition: A language L is in R_p if:

there exists polynomial p, and a deterministic polynomial time two-input algorithm M:

(i) If $x \in L$, $M[x,r]$ accepts for at least half the strings, r, of length $p(|x|)$.

(ii) If $x \notin L$, $M[x,r]$ rejects for each string r of length $p(|x|)$.

Let $|x|=n$, $p(n)=m$ and $W(x)=\{r \in \{0,1\}^m \mid M[x,r] \text{ accepts}\}$. Clearly, if $x \in L$, $|W(x)| \geq 1/2 \cdot 2^m$. In this case, $W(x)$ is called the *witness set*, and its elements are called *witnesses*. If $x \notin L$, $|W(x)|=\varphi$.

Given a fair coin, it is easy to pick a witness: a random m-bit string is a witness with probability at least 1/2. To show that a language $L \in R_p$ is also in SR_p, we will show how to sample polynomially many m-bit strings using a slightly-random source, such that if $x \in L$ then at least one of the sampled strings is

a witness with probability at least 1/2. If $x \notin L$ then clearly all the sampled strings will be non-witnesses. The witness set depends in general on the algorithm M and the input x. Since we want our sampling algorithm to work for every witness set, we shall show that even if the adversary picks the witness set *after* we specify our sampling algorithm, the algorithm will find a witness with probability at least 1/2.

4. The Algorithm

A slightly-random polynomial time algorithm must give the correct answer with high probability, even though the adversary, having complete knowledge of the algorithm and having infinite computational power, tries to make the algorithm err. To better illustrate the power of the adversary, we first show how he can thwart the obvious attempts to simulate an R_p algorithm.

Of course, the obvious simulation would convert the adversary's 'low-quality' bit sequence into a 'high-quality' one. This is ruled out by the theorem of Santha and Vazirani [SV] (see section 2) that in fact, it is impossible to get even one 'high-quality' bit from a slightly-random source.

What happens if we simply run the R_p algorithm without modification, substituting the slightly-random string, s, for the flips of a fair coin. Alon and Rabin [AR] show that if $W(x)$ is a random set, i.e. each m-bit string is a witness with probability 1/2, then $Pr[M[x,s] \text{ accepts}] \geq c$, where constant c depends on δ. By re-running the algorithm a constant number of times, this success probability can be increased to 1/2. However, if the adversary is allowed to choose $W(x)$, then he can ensure $Pr[M[x,s] \text{ accepts}] \leq \frac{1}{(const.)^m}$. Thus exponentially many runs of the algorithm would be necessary to reduce the error probability to 1/2. A simple example of this is the following: let $W(x) = \{r \mid |r| = m$, and majority of the bits of r are 1's$\}$. Let the adversary bias each bit of s towards 0 (i.e. $Pr[0] = 1-\delta$). Then the number of 1's in the binary representation of s is a random variable with mean δm and standard deviation $O(\sqrt{m})$. Thus the probability that $s \in W(x)$ is exponentially small.

As a final example consider the following algorithm: the m-bit slightly-random string s is mapped in polynomially many different ways: $f_1(s)$, ..., $f_{poly(m)}(s)$ in an attempt to find a witness. Once again the adversary has a strategy that limits the probability of success to $\frac{1}{(const.)^m}$. Let $Z = \{z \in \{0,1\}^m \mid z$ has at most $2\delta m$ 1's in its binary representation$\}$. Let $W(x)$ contain $\{f_i(z) \mid z \in Z\}$. Let the adversary bias each bit of s towards 0 (i.e. $Pr[0] = 1-\delta$). Then the probability that one of the mappings of s is a witness is equal to the probability that $s \in Z$, which is exponentially small.

Theorem 1: $R_p = SR_p$.

We will show that for any language $L \in R_p$, L is in SR_p. Let $M[x,r]$ be a random polynomial time algorithm that accepts L. The following algorithm, SAMPLE, uses the slightly-random source to generate polynomially many strings $r \in \{0,1\}^m$, and simulates $M[x,r]$, for each string. We will prove that if $x \in L$ then one of these strings must be a witness with probability $\Omega(\frac{1}{loglog\, m})$.

Procedure SAMPLE: On input x;

Let $m = p(|x|)$.

Let $k = 2(1-\delta)\log m$.

Flip the adversary controlled coin $m(k+1)$ times to obtain $k+1$ strings $s_1, s_2, \ldots, s_k, s$ each m bits long.

For each subset $S \in \{s_1, \ldots, s_k\}$ do:

if $M[x, s + \sum_{s_i \in S} s_i]$ accepts, then accept and halt.

end;

reject and halt;

end SAMPLE;

Algorithm SAMPLE simulates M on $2^k = m^{2(1-\delta)}$ inputs. Hence its running time is polynomial in m and therefore in $|x|$. The string s plays the same role as $s_1 \ldots s_k$; it has been singled out just to make the proof conceptually simpler. We will view each component of a string as an element of $GF[2]$. By the sum of two strings we mean component by component addition.

Notice that the adversary can tailor his strategy to any particular sampled string, say $r = s_1 + s_5 + s$, to force $Pr[r$ is a witness] to be exponentially small. However, we will show that there is no adversary strategy that simultaneously forces this probability to be small for each of the polynomially many strings generated by the algorithm, no matter what the witness set is.

Consider the constraints on s to ensure a successful run of the algorithm for the adversary: the apriori constraint is that s must be a non-witness. i.e. $s \in A_0 = \{0,1\}^m - W(x)$. Once s_1 has been generated by the adversary, s is further constrained: now s must lie in the set $A_1 = \{r \in \{0,1\}^m \mid r \in A_0$ and $r+s \in A_0\}$. Let A_i denote the subset of $\{0,1\}^m$ to which s is constrained after $s_1, \ldots, s_i$ have been generated. Then $A_0 \supseteq A_1 \supseteq \ldots\ldots \supseteq A_k$. We will show that even for the optimal distribution of the witnesses by the adversary, and the optimal strategy for biasing the coin, $|A_k| = 1$ with probability $\Omega(\frac{1}{loglog\, m})$, i.e. with probability $\Omega(\frac{1}{loglog\, m})$ one of the sampled points will be a witness.

Definition: $A_i = \{r \in \{0,1\}^m \mid$ for every subset S of $\{s_1,...,s_i\}$, $r + \sum_{s_j \in S} s_j \in A_0\}$.

The following lemma establishes an equivalent recursive definition of A_i.

Lemma 1: $A_{i+1} = \{r \in A_i \mid r + s_{i+1} \in A_i\}$, for $1 \leq i < k$.

Proof: The proof is by an induction on i. To prove the induction step, we first observe that $A_{i+1} \subseteq A_i$. Now, the set of witnesses *w.r.t.* $s_1...s_i$, i.e. $\{0,1\}^n - A_i$, consists of exactly those strings which map into $W(x)$ with some linear combination of $s_1...s_i$. So, a string $r \in A_i$ will map into $W(x)$ with some linear combination of $s_i...s_{i+1}$ *iff* $r + s_{i+1}$ is in $\{0,1\}^n - A_i$. Hence the recursive definition.

The above lemma defines the additional constraints imposed by s_{i+1} on s over and above the constraints imposed on it by $s_1,...,s_i$. We shall use this inductive characterization below to show that if $|A_i| = 2^{n-j}$ then the expected size of A_{i+1} is less than $2^{n-j/(1-\delta)}$ even for the adversary's optimal strategy. Thus $|A_i|$ decreases double exponentially in i.

Why is the expected size of A_{i+1} so much smaller than the size of A_i? First consider the simple case when s_{i+1} is generated by the flips of a fair coin. Let r be any m-length string. Then $r + s_{i+1}$ is equally likely to be any m-length string.

Let $i_r = \begin{cases} 1 & \text{if } r + s_{i+1} \in A_i \\ 0 & \textit{otherwise} \end{cases}$

Then $E(i_r) = \frac{|A_i|}{2^n} = 2^{-j}$.

$E(|A_{i+1}|) = E[\sum_{r \in A_i} i_r] = \sum_{r \in A_i} E[i_r] = 2^{-j}|A_i| = 2^{n-2j}$.

Thus the expected size of A_{i+1} is much smaller than the size of A_i if s_{i+1} is generated by the flips of a fair coin.

Let us return to the situation when s_{i+1} is generated by the adversary. Is it not conceivable that some clever choice of 2^{n-j} elements of $\{0,1\}^m$ for A_i and a clever adversary strategy for generating s_{i+1} might ensure that $(A_i + s_{i+1}) \cap A_i$ is almost all of A_i? Lemma 2 proves that no such clever strategy exists. We give an intuitive argument to motivate the lemma: the adversary would like $(A_i + s_{i+1})$ to be almost equal to A_i with high probability. Thus he would like to map almost all strings of A_i onto A_i, under mapping by s_{i+1}, with high probability. So A_i must be highly symmetrical. However, a symmetrical set A_i is bad for the adversary because then if he generates a bad string s_{i+1}, then *all* the elements of A_i would get mapped outside of A_i. Take for example $A_i = \{0r \mid r \in \{0,1\}^{m-1}\}$. If s_{i+1} has 0 in its first bit, $A_{i+1} = A_i$. However, with probability $\geq \delta$, s_{i+1} will have 1 in its first bit, and $A_{i+1} = \varphi$.

We first show that there is always an *extreme strategy* for the adversary, which maximizes the expected size of A_{i+1}. An extreme strategy is one in which the adversary biases its next bit to either δ or 1-δ, and no intermediate values. The bias is still a

function of the previously generated bits.

Lemma 2: Let $A \subseteq \{0,1\}^m$. Let

$A' = \{v \in A \mid v+r \in A\}$, where r is generated by the adversary.

There is an extreme strategy which maximizes $E(|A'|)$.

Proof: A strategy is a function from strings of length less than m to $[\delta, 1-\delta]$. Let F be the strategy that maximizes $E(|A'|)$, and is closest to being an extreme strategy; i.e. it takes on values strictly between δ and $1-\delta$ on the fewest points. Suppose $b_1 \cdots b_i$ is one such point where $\delta < F(b_1 \cdots b_i) < 1-\delta$. Then

$E(|A'|) = E(|A'| \mid$ first i bits are $b_1 \cdots b_i) +$

$E(|A'| \mid$ first i bits differ from $b_1 \cdots b_i)$.

$E(|A'| \mid$ first i bits are $b_1 \cdots b_i) = F(b_1 \cdots b_i)$

$E(|A'| \mid$ first $i+1$ bits are $b_1 \cdots b_i 0) + F(b_1 \cdots b_i)$

$E(|A'| \mid$ first $i+1$ bits are $b_1 \cdots b_i 1)$.

This is maximized at an extreme value of $F(b_1 \cdots b_i)$, contradicting the assumption that F was closest to being an extreme strategy.

Next we show that even for the best extreme strategy, $E(|A_i|)$ decreases rapidly:

Lemma 3: Let $A \subseteq \{0,1\}^m$: $\frac{|A|}{2^m} = f$. Let $A' = (A+s) \cap A$. Then $\frac{E(|A'|)}{2^m} \leq f^{\frac{1}{1-\delta}}$.

Proof: By an induction on m, we prove a stronger statement. We will consider two arbitrary sets $I\ T \subseteq \{0,1\}^m$, and will bound the expected size of H, $H = (I+s) \cap T$, where s is produced by the best extreme strategy of the adversary. The lemma will clearly follow from the following hypothesis:

Induction Hypothesis: Let $I, T \subseteq \{0,1\}^m$, $\frac{|I|}{2^m} = f$, $\frac{|T|}{2^m} = g$. Let $H = (I+s) \cap T$, where string s is a string produced by the best extreme strategy of the adversary. Then $\frac{E(|H|)}{2^m} \leq f^{\frac{1}{2(1-\delta)}} g^{\frac{1}{2(1-\delta)}}$.

Basis: The hypothesis is easily verified for $m = 1$.

Induction Step: Let $I, T \subseteq \{0,1\}^{m+1}$. $\frac{|I|}{2^{m+1}} = f$, $\frac{|T|}{2^{m+1}} = g$. Let $H = (I+s) \cap T$. We must show that $\frac{E(|H|)}{2^{m+1}} \leq f^{\frac{1}{2(1-\delta)}} g^{\frac{1}{2(1-\delta)}}$. Let $I = I_0 \cup I_1$, where $I_0 = \{r \in I$: first bit of r is 0$\}$, and $I_1 = \{r \in I$: first bit of r is 1$\}$. Similarly let $T = T_0 \cup T_1$. Let $\frac{|I_0|}{2^m} = f_0$, $\frac{|T_0|}{2^m} = g_0$. Then $\frac{|I_1|}{2^m} = 2f - f_0$, $\frac{|T_1|}{2^m} = 2g - g_0$,

$E(|H|) = E(|H| \mid$ first bit of s is 0$)\ Pr[$first bit of s is 0$] + E(|H| \mid$ first bit of s is 1$)\ Pr[$first bit of s is 1$]$.

W.l.o.g. assume that the first bit of s is 0 with probability $1-\delta$.

Using the induction hypothesis, and substituting $a = \frac{1}{2(1-\delta)}$, we get

$$E(|H|) \leq 2^{n-1}\{(1-\delta)[f_0^a g_0^a + (2f-f_0)^a (2g-g_0)^a] +$$

$$\delta[f_0^a (2g-g_0)^a + (2f-f_0)^a g_0^a]\}.$$

$(1-\delta)[x^a y^a + (2-x)^a(2-y)^a] +$
$\delta[x^a(2-y)^a + (2-x)^a y^a] \le 2.$

In Lemma 4 we will show that the left hand side attains its maximum at $x=y=1$ thus proving the inequality and completing the induction step.

Proof of Theorem 1: Let $A_0 \subseteq \{0,1\}^m$, $|A_0| = 2^{m-1}$ be an arbitrary set of non-witnesses. We will first give a simple proof to show that for $k = 0(\log m)$, $|A_k| \le 1$ with at least inverse polynomial probability, for any strategy of the adversary. We will then give a more precise argument to prove that for $k = 2(1-\delta)\frac{}{\delta}\log m$, $Pr[|A_k| \le 1] \ge \Omega(\frac{1}{\log\log m})$.

By lemma 3, if $|A| = 2^{m-i}$, $0 < i \le m$, $E(|A'|) \le 2^{m-2ai}$, where $2a = \frac{1}{1-\delta}$.

Let $Pr[|A'| \ge 2^{m-(a+1/2)i}] = p$.

Then $2^{m-(a+1/2)i}p + 0(1-p) \le 2^{m-2ai}$.

Therefore, $p \le 2\frac{1}{2^{(a-1/2)i}}$.

For our purpose, $i \ge 1$. So, choosing k *s.t.* $(a+1/2)^k = m$, we get

$Pr[|A_k| \le 1] \ge [1-\frac{1}{2^{(a-1/2)}}]^k$, which is inverse polynomial.

To get a better bound, we will decrease $|A_i|$ in two stages. Choose j *s.t.* $2^{(a-1/2)j} = \log m$, i.e. $j = \frac{1}{a-1/2}\log\log m$.

Let k_1 be *s.t.* $(a+1/2)^{k_1} = j$. Clearly, $k_1 = 0(\log\log\log m)$.

Now, $Pr[|A_{k_1}| \le 2^{m-j}] \ge [1-\frac{1}{2(a-1/2)}]^k$,
$\ge \Omega(\frac{(1}{\log\log m})$.

For the second stage, choose k_2 *s.t.* $j(a+1/2)^{k_2} = m$. Let $k = k_1+k_2$. Clearly, $k \le 2\frac{(1-\delta)}{\delta}\log m$.

$Pr[|A_k| \le 1 \mid |A_{k_1}| \le 2^{m-j}] \ge (1-\frac{1}{\log m})^{\log m}$
$\ge const.$

Therefore, $Pr[|A_k| \le 1] \ge \Omega(\frac{1}{\log\log m})$.

Once $|A_k| \le 1$, the probability that the adversary can generate $s \in A_k$ is at most $(1-\delta)^n$, i.e. inverse exponential. So with probability $\Omega(\frac{1}{\log\log m})$, at least one of the 2^k strings will be a witness. This is true for each run of the algorithm, no matter what the outcome of the previous runs. So, by running this algorithm polynomially many times, the probability of finding a witness can be increased to 1/2.

Lemma 4: The function

$f(x,y) = (1-\delta)\ [x^a y^a + (2-x)^a(2-y)^a] +$
$\delta[x^a(2-y)^a + (2-x)^a y^a]$

attains its maximum at $x=y=1$, under the following conditions $0 \le x,y \le 2$, $a = \frac{1}{2(1-\delta)}$, $0 < \delta < 1/2$.

Proof: We will first disregard the relation $a = \frac{1}{2(1-\delta)}$, and show that under the conditions

$0 \le x,y \le 2,\ 0 < \delta < 1/2,\ 1/2 < a < 1,$
$\frac{\delta f}{\delta x} = \frac{\delta f}{\delta y} = 0$ only at $x=y$.

$$\frac{1}{a}\frac{\delta f}{\delta x} = (1-\delta)[x^{a-1}y^a - (2-x)^{a-1}(2-y)^a] + \delta[x^{a-1}(2-y)^a - (2-x)^{a-1}y^a]$$

$$\frac{1}{a}\frac{\delta f}{\delta y} = (1-\delta)[x^a y^{a-1} - (2-x)^a(2-y)^{a-1}] + \delta[(2x-x)^a y^{a-1} - x^a(2-y)^{a-1}].$$

To ensure $\frac{\delta f}{\delta x} = \frac{\delta f}{\delta y}$, we have

$$\frac{x^{a-1}y^a-(2-x)^{a-1}(2-y)^a}{x^{a-1}(2-y)^a-(2-x)^{a-1}y^a} = \frac{x^a y^{a-1}-(2-x)^a(2-y)^{a-1}}{(2-x)^a y^{a-1}-x^a(2-y)^{a-1}}$$

$$\Rightarrow (2-x)^{a-1}x^{a-1}[y^{2a-1}+(2-y)^{2a-1}] = y^{a-1}(2-y)^{a-1}[x^{2a-1}+(2-x)^{2a-1}]$$

$$\Rightarrow y^a(2-y)^{1-a}+y^{1-a}(2-y)^a = x^a(2-x)^{1-a}+(2-x)^a x^{1-a}.$$

This equation holds for $x=y$ and $x=2-y$. We will show that these are the only solutions. Let

$$g(x)=x^a(2-x)^{1-a}+(2-x)^a x^{1-a}.$$

It is easy to check that $\frac{dg}{dx}$ is positive for $x<1$ and negative for $x>1$. So, $g(x)$ is an increasing function in the range $0<x<1$, and a decreasing function in the range $1<x<2$. Therefore, for each value of b, $g(x)=b$ can have at most two solutions. Hence the equation holds only for $x=y$ and $x=2-y$.

It is easy to check that $\frac{\delta f}{\delta x}>0$ and $\frac{\delta f}{\delta x}<0$ in the quadrant $2-x>x$, $2-y<y$, and that $\frac{\delta f}{\delta x}<0$ and $\frac{\delta f}{\delta y}>0$ in the quadrant $2-x<x$, $2-y>y$. Therefore $\frac{\delta f}{\delta x}=\frac{\delta f}{\delta y}=0$ only at $x=y$.

We will next show that $f(x,x) = (1-\delta)[x^{2a}+(2-x)^{2a}] + 2\delta x^a(2-x)^a$ attains its maximum at $x=1$, under the given conditions. We will first substitute $y = \left\{\frac{2-x}{x}\right\}^a$, i.e. $x = \frac{2}{1+y^{1/a}}$, to obtain $h(y) = 2^{2a}(1+y^{1/a})^{-2a}[(1-\delta)(1+y^2)+2\delta y]$. It is enough to show that $h(y)$ achieves its maximum at $y=1$, under the condition $y>0$.

$$\frac{1}{2^{2a}}\frac{dh}{dy} = 2(1+y^{1/a})^{-2a-1}[(1+y^{1/a})((1-\delta)y+\delta) - y^{1/a-1}((1-\delta)(1+y^2)+2\delta y)]$$

Now, it is sufficient to show that the function

$$i(y) = (1+y^{1/a})[(1-\delta)y+\delta] - y^{1/a+1}[(1-\delta)(1+y^2)+2\delta y]$$

satisfies: $i(y)>0$ for $0<y<1$

$i(y)=0$ for $y=1$

$i(y)<0$ for $y>1$.

First we simplify $i(y)$,

$$i(y) = 2\delta + (1-\delta)y - (1-\delta)y^{1/a-1} - \delta y^{1/a}$$

$$\frac{di}{dy} = (1-\delta) - \frac{1}{a}\delta y^{1/a-1} - (\frac{1}{a}-1)(1-\delta)y^{1/a-2}.$$

$$\frac{d^2i}{dy^2} = -\frac{1}{a}(\frac{1}{a}-1)\delta y^{1/a-2} - (\frac{1}{a}-1)(\frac{1}{a}-2)(1-\delta)y^{1/a-3}.$$

$$= (\frac{1}{a}-1)y^{1/a-3}\frac{1}{a}[-\delta y + (2a-1)(1-\delta)].$$

Substituting $(1-\delta) = \frac{1}{2a}$,

$$\frac{d^2i}{dy^2} = (\frac{1}{a}-1)y^{1/a-3}\frac{1}{a}(1-\frac{1}{2a})(1-y).$$

Since $1/2<a<1$, $\frac{d^2i}{dy^2}$ is

<0 for $0<y<1$
$=0$ for $y=1$
>0 for $y>1$.

These conditions, together with the observation that $\frac{di}{dy} = 0$ at $y=1$, imply that $\frac{di}{dy} < 0$ for $0<y<1$ and

$y>1$. Hence i is a decreasing function in the range $0<y<1$ and $y>1$. On the other hand $i(1)=0$. Hence $i(y)$ satisfies the required conditions, and $f(x,x)$ attains its maximum at $x=1$. It is easy to check that $f(x,y)\leq 2$ at the boundaries of the given region. Since $f(1,1)=2$, $f(x,y)\leq 2$ under the given conditions.

5. Extensions

Solving a *BPP* problem with a slightly-random source is much harder, because we must ensure that a *majority* of the query strings hit witnesses with probability $1/2+\epsilon$, $\epsilon>0$. This needs a different argument, and will be presented in a forthcoming paper. We will also extend this argument to the problem of sampling a population using a slightly-random source.

Let W be a population, i.e. a set of elements. Each element is either 0 or 1. Let p be the fraction of elements that are 0. We want to sample W, using a slightly-random source, and estimate p with a certain accuracy and confidence level. We will assume here that the elements of W have been numbered from 1 to n, where $n = |W|$. Also, for each i, $1 \leq i \leq n$, we can determine the i^{th} element. We will show that using a slightly-random source having bias in the range $[\delta, 1-\delta]$ and sampling W at $c_1 \log^{c_2} n$ points, we can estimate p. The constant c_1 depends on the desired accuracy and confidence level and c_2 depends on δ.

6. Open Problem

Notice that the conversion from an R_p to an SR_p algorithm increases the running time by a polynomial factor, whose degree depends on δ. However, in the special case of testing polynomial identities [BV], the polynomial factor has fixed degree, independent of δ. Can a similar SR_p algorithm be obtained for primality testing?

Acknowledgements

We are grateful, especially to Manuel Blum for providing new insights at critical moments, and to Dick Karp and Michael Friedman for several valuable discussions on the nature and the power of the adversary. Finally, we wish to thank Rakesh, Mathematics Department, Cornell University, for generously sharing his time and skill in dealing with complicated inequalities, leading to the proof of Lemma 5, Gilles Brassard and Sampath Kannan for many interesting discussions.

References

[Ad] L. Adleman, "Two Theorems on Random Polynomial Time," 19th FOCS (1978), 75-83.

[AR] N. Alon and M.O. Rabin, "On the Random Properties of a Weakly Random Source," to appear.

[AW] M. Ajtai and A. Wigderson, "Deterministic Simulation of Probabilistic Constant Depth Circuits," this conference proceeding.

[Bl] M. Blum, "Independent Unbiased Coin Flips from a Correlated Biased Source: A Finite State Markov Chain," 29th STOC, (1984), 425-433.

[BR] K. Baclawski and G. Rota, *An Introduction Probability and Random Processes.*

[BV] M. Blum and U.V. Vazirani, "An SR_p Algorithm for Verification of Polynomial Identities," to appear.

[CG] B. Chor and O. Goldreich, "Unbiased Bits from Weak Sources of Randomness," this conference.

[Gi] J. Gill, "Computational Complexity of Probabilistic Turing Machines," SIAM J. Comput. 6 (1977) pp. 675-695.

[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).

[Mi] G.L. Miller, Riemann's hypothesis and tests for primality, J. Comput. System Science. 13 (1976), 300-317.

[Ra1] M.O. Rabin, "Probabilistic Algorithms," edited by J.F. Traub, Algorithms and Complexity, Academic Press, New York (1976) pp. 21-39.

[Ra2] M.O. Rabin, "Probabilistic Algorithms in Testing Primality," J. Number Theory 12 (1980) pp. 128-138.

[Si] M. Sipser, "A Complexity Theoretic Approach to Randomness," 15th STOC, (1983), 330-335.

[SS] R. Solovay and V. Strassen, "A Fast Monte-Carlo Test for Primality," SIAM J. Comput. 6 (1977) pp. 84-85.

[St] L. Stockmeyer, "The Complexity of Approximate Counting," 15th STOC, (1983), 118-126.

[SV] M. Santha and U.V. Vazirani, "Generating Quasi-random Sequences from Slightly-random Sources," Proc. 25th Ann. Symp. on the Theory of Computing. Oct. 1984, pp. 434-440.

[Va] U.V. Vazirani, Towards a Strong Communication Complexity Theory or Generating Quasi-Random Sequences from Two Communicating Slightly-random Sources," to appear in JCSS (currently, Proc. STOC 1985).

[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," 23rd FOCS, (1982), 80- 91.