Electronic Colloquium on Computational Complexity, Report No. 167 (2011)

Improving on Gutfreund, Shaltiel, and Ta-Shma's paper "If NP Languages are Hard on the Worst-Case, Then it is Easy to Find Their Hard Instances"

Nikolay Vereshchagin*†

Abstract

Assume that NP $\neq$ RP. Gutfreund, Shaltiel, and Ta-Shma in [Computational Complexity 16(4):412-441 (2007)] have proved that for every randomized polynomial time decision algorithm D for SAT there is a polynomial time samplable distribution such that D errs with probability at least $1/6 - \varepsilon$ on a random formula chosen with respect to that distribution. In this paper, we show how to increase the error probability to $1/3 - \varepsilon$. We also generalize this result to the search version of SAT: we prove that for every randomized polynomial time algorithm S searching for a satisfying assignment of a given formula, there is a polynomial time samplable distribution such that S errs with probability at least $1 - \varepsilon$ on a random formula chosen with respect to that distribution.

*The work was in part supported by the RFBR grant 09-01-00709 and the ANR grant ProjetANR-08-EMER-008

†Moscow State University, Leninskie gory 1, Moscow 119992, Russia, E-mail: ver@mccme.ru, WWW home page: http://lpcs.math.msu.su/~ver

ISSN 1433-8092

1 Introduction

A goal of the Average Case Complexity is to show that one way functions exist under a worst case hardness assumption like NP $\neq$ RP.[1] Or, at least to show that SATISFIABILITY (SAT) is hard on average. The latter can be understood in two quite different ways:

(1) There is a sampler G such that the following holds: For every probabilistic polynomial time algorithm S, for infinitely many n with probability close to 1, the formula produced by $G(1^n)$ is satisfiable but S does not find its satisfying assignment. (The probability is with respect to the product of the uniform distribution over G's internal tosses and the uniform distribution over S's internal tosses.)

(2) There is a sampler G such that the following holds: For every probabilistic polynomial time decision algorithm D, for infinitely many n with probability close to $1/2$, D errs on the formula ψ produced by $G(1^n)$ (which means that D answers YES while ψ is not satisfiable or vice verse).

Let us define samplers; in this paper, we use the framework of Bogdanov and Trevisan [2] rather than the original Levin's one from [4].

Definition 1. A sampler is a polynomial time probabilistic algorithm G that given 1^n as input outputs a Boolean formula of length *at least* n. If the length of the output formula is always exactly n, we call the sampler *proper*.

In this paper, we consider Boolean formula in the basis $\neg, \vee, \wedge, 0, 1$. The length $|\varphi|$ of a formula φ is defined as the number of symbols in it: every variable is counted as one symbol. Actually, in [2] samplers generate binary strings and not formulas, and samplers are always proper. To generate formulas, it is more natural to consider non-proper samplers, because this makes the results encoding-invariant.

The goal (1) is certainly closer, than the second one, to the initial goal of constructing one way functions. More specifically, existence of "infinitely often" (i.o.) one way functions is equivalent to constructing a sampler as in (1), which generates a satisfiable formula ψ *together with its satisfying assignment* a. (The projection $(\psi, a) \mapsto \psi$ would be the sought i.o. one-way function.)

The paper [3] makes a step towards the goal (2). Namely, [3] shows that the assumption NP $\neq$ RP implies the following weaker version of (2), in which we *allow the sampler G depend on the decision algorithm D*:

[1] NP $\neq$ RP means that there is no polynomial time randomized algorithm that given any satisfiable Boolean formula with probability at least $1/2$ finds its satisfying assignment.

(2') For every probabilistic polynomial time algorithm D there is a *proper* sampler G such that the following holds: for almost all n with probability at least 0.03, D errs on the formula ψ produced by $G(1^n)$. Actually, [3] requires the syntax of formulas allow padding: given a formula φ one can find in polynomial time a formula of length $|\varphi| + 1$, which is equivalent to φ.

The authors of [3] remark that they do not optimise constants and by careful calculation the constant 0.03 can be improved; however they do not see how to get it above 1/3. Indeed, a careful calculation shows that 0.03 may be replaced by $1/24 - \varepsilon$ (for any positive ε). But we do not see how, using the techniques of [3], to get the constant above 1/24.

In this paper:

- We show that one can generalize the results of [3] to the search version of SAT. That is, we show that for every polynomial time probabilistic algorithm S and ε there is a sampler G such that for infinitely many n, with probability at least $1 - \varepsilon$ the following holds: the formula φ produced by $G(1^n)$ is satisfiable, however $S(\varphi)$ does not find its satisfying assignment. This is our first result.

  In this result, it is important that the length of the formula produced by $G(1^n)$ tends to infinity, as n tends to infinity. Otherwise, the statement would become trivial, as G might produce any fixed formula on which S errs with high probability. The same applies to the other results in this paper.

- We notice that, for non-proper samplers, the constant 0.03 in (2') can be improved to $1/6 - \varepsilon$.

- We show that, using an extra trick, one can get even $1/3 - \varepsilon$ (for non-proper samplers). This is our second result. The question whether one can replace 1/3 by 1/2 remains open.

In other words, we show how to double the error probability in (2') and prove a version (1') of (1), in which sampler G is allowed to depend on the algorithm S.

2 Generating hard instances of search version of SAT

We start with explaining the main idea of [3] so that it be clear what our contribution is.

Definition 2. The *search version of SAT* is the following problem: given a Boolean formula φ find its satisfying assignment. A (randomized) *SAT solver* is a (randomized) polynomial time algorithm that for every input formula φ either finds its satisfying assignment, or says "don't know". A SAT solver D *errs* on ψ if ψ is satisfiable and $D(\psi) =$"don't know".

Theorem 1 ([3])**.** *Assume that NP $\neq$ P. Given a deterministic SAT solver S one can construct a deterministic polynomial time procedure that given 1^n produces a formula ψ_n of length at least n such that S errs on ψ_n for infinitely many n.*

Proof. Consider the following search problem in NP.
Search problem P:
Instance: a string 1^n over the unary alphabet.
Solution: a pair (ψ, a) where ψ is a satisfiable formula of length n such that $S(\psi) =$"don't know", and a is its satisfying assignment.

We will call an instance 1^n of P *solvable* if such pair (ψ, a) exists. As SAT is NP complete, the search problem P reduces to the search version of SAT. This means that there is a polynomial time algorithm that given 1^n finds a formula, called φ_n, such that:
(1) if the instance 1^n of P is solvable then φ_n is satisfiable, and
(2) given any satisfying assignment of φ_n we can find (in polynomial time) a solution to the instance 1^n.
The length of φ_n is bounded by a polynomial n^d and w.l.o.g. we may assume that $|\varphi_n| \geq n$.

The procedure works as follows: given 1^n, as input
(a) find the formula φ_n;
(b) run $S(\varphi_n)$;
(c) if $S(\varphi_n)$="don't know"then output φ_n and halt;
(d) otherwise $S(\varphi_n)$ produces a satisfying assignment for φ_n; given that assignment find (in polynomial time) a solution (ψ, a) to the instance 1^n of the problem P; output ψ and halt.

Since we assume that P $\neq$ NP, for infinitely many n the instance 1^n of P is solvable. For such n either $S(\varphi_n)$="don't know"(and thus S errs on φ_n), or (ψ, a) is a solution to 1^n (and thus S errs on ψ). □

The next idea of [3] is to generalize this theorem to randomized SAT solvers. This is done as follows. Let S be a randomized SAT solver working in time n^c and let r be string of length at least n^c. We will denote by S_r the algorithm S that uses bits of r as coin flips.

Theorem 2 ([3]). *Assume that NP $\neq$ RP. Then for some natural constant d the following holds. Let S be a randomized SAT solver and let n^c denote its running time on formulas of length n. Then there is a deterministic polynomial time procedure that given any binary string r of length $n^{c^2 d}$ produces a formula η_r of length between n and n^{cd}, where for any positive ε for infinitely many n the following holds. For a fraction at least $1-\varepsilon$ of r's the algorithm S_r errs on η_r.*

Notice that the length of η_r is at most n^{cd}. Therefore the running time of S for input η_r is at most $n^{c^2 d}$. Hence $S_r(\eta_r)$ is well defined.

Proof. The proof is very similar to that of the previous theorem. The only change is that we have to replace the search problem P by the following problem P':
Instance: a binary string r' of length n^c (for some n).
Solution: a satisfiable formula ψ of length n and its satisfying assignment a such that $S_{r'}(\psi) =$"don't know".

Let $r' \mapsto \varphi_{r'}$ be a reduction of P' to the search version of SAT. The length of $\varphi_{r'}$ is bounded by a polynomial n^{cd} of $|r'| = n^c$ and w.l.o.g. we may assume that $|\varphi_{r'}| \geq n$.

The procedure required in the theorem, called **Procedure A**, works as follows: given r of length $n^{c^2 d}$, as input,
(a) let r' stand for the prefix of r of length n^c;
(b) find the formula $\varphi_{r'}$;
(c) run $S_r(\varphi_{r'})$;
(d) if $S_r(\varphi_{r'})$="don't know"then output $\varphi_{r'}$ and halt;
(e) otherwise $S_r(\varphi_{r'})$ produces a satisfying assignment for $\varphi_{r'}$; given that assignment find (in polynomial time) a solution (ψ, a) to the instance r' of the problem P'; output ψ and halt. (End of Procedure A.)

Let η_r stand for the formula output by the procedure. Since we assume that NP $\neq$ RP, for every positive ε the randomized searching algorithm S errs with probability at least $1-\varepsilon$ for infinitely many input formulas. This implies that for infinitely many n the number of solvable instances r' of the problem P' is at least $(1-\varepsilon)2^{n^c}$. For those r's the formula $\varphi_{r'}$ is satisfiable. Therefore, for all but a fraction ε of r's the algorithm S_r errs on $\varphi_{r'}$ or $S_{r'}$ errs on ψ, which implies that S_r errs on ψ as well. □

Remark 1. Theorem 2 holds for $\varepsilon = 1/n^k$ for any constant k. Indeed, the assumption NP $\neq$ RP implies that the randomized searching algorithm S errs with probability at least $1-|\varphi|^{-k}$ for infinitely many input formulas φ.

With this machinery at hand we are able to prove our first result.

Theorem 3. *If NP $\neq$ RP then for every probabilistic polynomial time search algorithm S and for every positive ε there is a sampler G such that for infinitely many n the algorithm S errs on the formula produced by $G(1^n)$ with probability at least $1-\varepsilon$.*

Proof. Let us try first the following sampler G: for input 1^n toss a coin n^{c^2d} (where n^c is S's running time for input formulas of length n and d is the constant from Theorem 2), then apply Procedure A from Theorem 2 to r. Unfortunately, this sampler is poor, as S_r errs on η_r only and might not err on η_s for $s \neq r$. Thus we can show only that S errs on η_r with probability at least $(1-\varepsilon)2^{-n^{c^2d}}$.

Roughly speaking, this problem can be handled as follows. Let K stand for the number of formulas of length between n and n^{cd}. On average, the same formula ψ appears $2^{n^{c^2d}}/K$ times as η_r. If each formula appeared exactly $2^{n^{c^2d}}/K$ times, then our sampler would produce the uniform distribution and the probability of error of S on the generated random formula would be close to $1/K$. Assume now that we have applied Theorem 2 to the amplified version $\bar{S}$ of S in place of S (the algorithm $\bar{S}$ applies S for the same input, say, t times). The probability of error of $\bar{S}$ for every particular formula is the tth power of that of S. Hence the probability of error of S on the generated formula would be close to $(1/K)^{1/t}$. Choosing an appropriate polynomial $t(n)$ we can get this number close to 1.

In what follows we make this reasoning precise. More or less we show that the analysed case (when the sampler generates the uniform distribution) is the worst one.

Let $N(n)$ stand for the number of formulas of length n. Choose a polynomial $t(n)$ so that $(1/N(n))^{1/t(n)}$ be greater than, say, $1-\varepsilon$. As $N(n)$ is bounded by an exponent of n, such polynomial $t(n)$ does exist. Consider the following randomized SAT solver $\bar{S}$: given a formula φ, as input, invoke $t(|\varphi|)$ times the algorithm $S(\varphi)$; if at least one of the results is different from "don't know", then output it and halt; otherwise halt with "don't know" result.

Let n^c stand for the running time of $\bar{S}$. The sampler G works as follows: given 1^n, as input, toss a fair coin n^{c^2d} times to get a random string r of length n^{c^2d} and output the formula η_r produced by the procedure A from Theorem 2 applied to $\bar{S}$ in place of S.

Let p_n denote the probability distribution generated by $G(1^n)$:

$$p_n(\psi) = P[G(1^n) = \psi] = \frac{\#\{r \mid \eta_r = \psi\}}{2^{n^{c^2d}}}$$

(where $P[A]$ denotes the probability if the event A). By Theorem 2 only formulas of lengths between n and n^{cd} may have positive probability.

An analysis of G's work. Obviously for every formula ψ,

$$P[S(\psi) \text{ errs}] = P[\bar{S}(\psi) \text{ errs}]^{1/t(|\psi|)}.$$

Thus the error probability of S for a random input formula generated by sampler $G(1^n)$ equals

$$\sum_{i=n}^{n^{cd}} \sum_{|\psi|=i} p_n(\psi)\, P[\bar{S}(\psi) \text{ errs}]^{1/t(i)}.$$

Let $q_n(\psi)$ stand for the ratio

$$\frac{\#\{r \mid \eta_r = \psi,\ \bar{S}_r(\psi) \text{ errs}\}}{2^{n^{c^2 d}}}.$$

By Theorem 2 for infinitely many n for all but a fraction ε of r's the algorithm $\bar{S}_r$ errs for input formula η_r. This means that for infinitely many n we have

$$\sum_{\psi} q_n(\psi) = \frac{\#\{r \mid \bar{S}_r(\eta_r) \text{ errs}\}}{2^{n^{c^2 d}}} \geq 1 - \varepsilon. \tag{1}$$

On the other hand, $q_n(\psi)$ is a lower bound for both $P[\bar{S}(\psi) \text{ errs}]$ and $p_n(\psi)$. Therefore, the error probability of S on the random formula produced by $G(1^n)$ is at least

$$\sum_{i=n}^{n^{cd}} \sum_{|\psi|=i} q_n(\psi)^{1+1/t(i)}. \tag{2}$$

We claim that the inequality (1) together with the choice of the polynomial $t(n)$ imply that the sum (2) is at least $1 - 3\varepsilon$ for all large enough n. Indeed, by the convexity of the function $x \mapsto x^\alpha$ (for $\alpha > 1$) for all $a_1, \ldots, a_k$ we have

$$a_1^\alpha + \cdots + a_k^\alpha \geq k\Big(\frac{a_1 + \cdots + a_k}{k}\Big)^\alpha = (a_1 + \cdots + a_k)^\alpha (1/k)^{\alpha-1}.$$

In other words, the sum $a_1^\alpha + \cdots + a_k^\alpha$ is minimal when all the summands are equal.

Applying this inequality to the inner sum in (2) we see that it is at least

$$\Big(\sum_{|\psi|=i} q_n(\psi)\Big)^{1+1/t(i)} (1/N(i))^{1/t(i)}$$

Here $N(i)$ stands for the number of formulas of length i and by the choice of the polynomial $t(i)$ the second factor in the displayed inequality is at least $1 - 1/i$. Therefore the inner sum in (2) is greater than

$$(1 - 1/i)\Big(\sum_{|\psi|=i} q_n(\psi)\Big)^{1+1/t(i)},$$

which is more than

$$(1 - \varepsilon)\Big(\sum_{|\psi|=i} q_n(\psi)\Big)^{1+1/t(n)}$$

for all large enough n. Applying the convexity inequality to the outer sum in (2) we see that it is at least

$$(1 - \varepsilon)\Big(\sum_{\psi} q_n(\psi)\Big)^{1+1/t(n)} \Big(\frac{1}{n^{cd} - n + 1}\Big)^{1/t(n)}$$

Recall that $\sum_\psi q_n(\psi) \geq 1 - \varepsilon$ for infinitely many n. Thus the sum (2) is at least

$$(1 - \varepsilon)(1 - \varepsilon)^{1+1/t(n)} \Big(\frac{1}{n^{cd} - n + 1}\Big)^{1/t(n)}.$$

The term $\left(\frac{1-\varepsilon}{n^{cd}-n+1}\right)^{1/t(n)}$ tends to 1, as n tends to infinity. Hence for infinitely many n the sum (2) is at least $(1-\varepsilon)(1-\varepsilon)^2$. Hence the probability that S errs on the formula produced by $G(1^n)$ is at least $(1 - \varepsilon)^3 \geq 1 - 3\varepsilon$. Starting with $\varepsilon/3$ in place of ε we obtain the result. □

Remark 2. One can easily verify that Theorem 3 holds for $\varepsilon = 1/n^k$ for any constant k.

3 Generating hard instances of the decision version of SAT

Again we start with the following result, which is implicit in [3], so that our contribution be clear. We say that a (randomized) decision algorithm D *errs on a formula* φ if $D(\varphi) =$YES and φ is not satisfiable or vice verse.

Theorem 4 ([3])**.** *If NP* $\neq$ *RP then for every probabilistic polynomial time decision algorithm* D *and every positive* ε *there is a sampler* G *such that for infinitely many* n *the decision algorithm* D *errs on the formula produced by* $G(1^n)$ *with probability at least* $1/6 - \varepsilon$*.*

Proof. Let D and ε be given. First we use the standard amplification, as in [1], to transform D into decision algorithm $\bar{D}$: given a formula φ of length n as input the algorithm $\bar{D}$ invokes $D(\varphi)$ polynomial number K of times and outputs the most frequent result. The number $K = K(n)$ should be so large that the probability of the following event be less than 2^{-n}: there is a formula ψ of length n such that the frequency of YES answers in the run of $\bar{D}(\psi)$ differs from the probability of the event $D(\psi)$=YES by at most ε. By Chernoff bound, there is such polynomial $K(n)$.

Using the standard binary search techniques we transform $\bar{D}$ to a SAT solver S. That is, given a formula φ the algorithm S first runs $\bar{D}(\varphi)$. If the result is YES then it substitutes both $x = 0$ and $x = 1$ for the first variable x in φ and runs $\bar{D}$ on the resulting formulas $\varphi_{x=0}$, $\varphi_{x=1}$. If at least one of these runs outputs YES, we replace φ by the corresponding formula and recurse. Otherwise we return "don't know"and halt.

If $\bar{D}$ returns NO for the input formula φ, we return "don't know"and halt. Finally, if we have substituted 0s and 1s for all variables and the resulting formula is true, we return the satisfying assignment we have found, and otherwise we return "don't know".

Let n^c be the upper bound of S's running time for input formulas of length n and let r be a string of length n^c used as randomness for S. In its run for input φ the algorithm S uses parts of r as coin flips for $\bar{D}$. The part used in ith call of $\bar{D}$ depends on the length n and on i. We will denote that part by $r(n, i)$. When n and i are clear from the context we will write $\bar{D}_r$ in place of $\bar{D}_{r(n,i)}$.

The heart of the construction is a procedure that given any formula ψ and randomness r such that S_r errs on ψ returns at most three formulas such that the algorithm D errs on at least one of these formulas with high probability.

Procedure B. Given an input formula ψ, run $S_r(\psi)$ to find the place in the binary search tree where S_r is stuck. By the construction of S this may happen in the following three cases:

(1) $\bar{D}_r(\psi)$=NO. In this case output ψ.

(2) $S_r(\psi)$ performs the binary search till the very end but the resulting formula η is false. In this case output η.

(3) In the remaining case $S_r(\psi)$ is stuck in the middle of the binary search and thus has found a formula φ and its variable x such that $\bar{D}_r(\varphi)$ =YES while both $\bar{D}_r(\varphi_{x=0})$ and $\bar{D}_r(\varphi_{x=1})$ are NO. In this case return $\varphi, \varphi_{x=0}, \varphi_{x=1}$. (End of Procedure B.)

Apply Theorem 2 to S. We obtain Procedure A that given a string r of length $n^{c^2 d}$ returns a formula η_r of length between n and n^{cd} such that for

infinitely many n, S_r errs on η_r (except for a fraction at most ε of r's).

The sampler G works as follows. For input 1^n choose a random string r of length n^{c^2d}. Then apply Procedure A to r to obtain η_r. Then apply Procedure B to S, r and η_r to obtain at most three formulas. Finally choose one of these formulas at random and output it.

We claim that for infinitely many n the algorithm D errs on the formula produced by $G(1^n)$ with probability close to 1/6. To prove this claim notice that $S_r(\eta_r)$ calls $\bar{D}$ at most $2n^{cd}$ times. Let $i \leq 2n^{cd}$ be a number of a call and φ a formula of length between n and n^{cd}. Call a string r of length n^{c^2d} *bad for the pair* (i, φ) if the frequency of YES answers of D for input φ in the run of $\bar{D}_{r(|\varphi|,i)}(\varphi)$ differs from the probability of the event $D(\varphi)$=YES by more than ε. Call a string r of length n^{c^2d} *bad for the pair* (i, l) if it is bad for a pair φ, i where the length of φ is l. By construction of D for every pair (i, l) a fraction at most 2^{-l} of r's are bad for (i, l). Call a string r of length n^{c^2d} *bad* if there is l between n and n^{cd} and a number $i \leq 2n^{cd}$ such that r is bad for i, l (and call r *good* otherwise). An easy calculation shows that r is bad with probability at most

$$\sum_{l=n}^{n^{cd}} 2n^{cd}2^{-l} < n^{cd}2^{-n+2}.$$

Notice that all formulas in the search tree of $S_r(\eta_r)$ have the same length, as that of η_r. Thus, if r is good and S_r errs on η_r then the error probability of D on the formula output by Procedure B is at least $1/3(1/2 - \varepsilon)$. We need to subtract from this number the probability that r is bad and also the probability that S_r does not err on η_r. The latter one is at most ε for infinitely many n, so we are done. □

Remark 3. For proper samplers the constant 1/6 should be reduced to 1/24 by the following reason. Using a padding we may assume that the formula output by the sampler constructed in Theorem 4 has length either n, or n^{cd} (and not in between). Consider a new sampler $\tilde{G}$ that runs $G(1^n)$ and $G(1^{n^{1/cd}})$ and if either of the runs produces a formula of length n, then we output that formula (if both runs produce a formula of length n then we output each of them with probability 1/2). This yields the constant $1/24-\varepsilon$. Indeed, assume that $G(1^m)$ produces a formula φ such that $D(\varphi)$ errs with probability $1/6-\varepsilon$. Then either the event "$D(\varphi)$ errs and the length of φ is m" or the event "$D(\varphi)$ errs and the length of φ is m^{cd}" has probability at least $1/12-\varepsilon/2$. In the first case the probability of the event "D errs on the

output of $G(1^m)$" is at least $1/24 - \varepsilon/4$. In the second case the probability of the event "D errs on the output of $G(1^{m^{cd}})$" is at least $1/24 - \varepsilon/4$.

Now we are able to present our second result, which improves the constant 1/6 to 1/3 in the above theorem.

Theorem 5. *If NP $\neq$ RP then for every probabilistic polynomial time decision algorithm D and every positive ε there is a sampler G such that for infinitely many n the decision algorithm D errs on the formula produced by $G(1^n)$ with probability at least $1/3 - \varepsilon$.*

Proof. The proof is similar to that of the previous theorem. We just change the final stage of the sampler G as follows. Instead of choosing one of the three formulas at random with equal probabilities, we select the output formula based on the frequencies of YES answer in the run of D_r on them. More specifically, let φ be a formula such that $\bar{D}(\varphi)$ =YES while both $\bar{D}_r(\varphi_{x=0})$ and $\bar{D}_r(\varphi_{x=1})$ are NO. Let u, v, w be the frequencies of the results YES, NO, NO in the runs of D_r on $\varphi, \varphi_{x=0}, \varphi_{x=1}$ which the algorithm $\bar{D}_r$ has observed. By assumption, all the numbers u, v, w are at least 1/2. If $u < 2/3$ we output φ and halt, as in this case the probabilities of both events $\bar{D}(\varphi)$ =NO, $\bar{D}(\varphi)$ =YES are greater than $1/3 - \varepsilon$ (assuming that r is good). We proceed similarly if v or w is less than 2/3.

Otherwise find non-negative rational numbers p, q, s such that $p+q+s = 1$ and all the numbers

$$pu+q(1-v)+s(1-w), \quad p(1-u)+qv+s(1-w), \quad p(1-u)+q(1-v)+sw \quad (3)$$

are at least 1/3 (we will argue later that such p, q, s exist). Then output $\varphi, \varphi_{x=0}, \varphi_{x=1}$ with probabilities p, q, s respectively and halt.

Obviously, at least one of the answers of

$$\bar{D}(\varphi) = \text{YES}, \quad \bar{D}(\varphi_{x=0}) = \text{NO}, \quad \bar{D}(\varphi_{x=1}) = \text{NO}$$

is wrong. If the answer $\bar{D}(\varphi)$=YES is wrong, that is, the formula φ is not satisfiable, then the probability that D errs on φ is at least $u - \varepsilon$. The probability that D errs on $\varphi_{x=0}$ is at least $1 - v - \varepsilon$. The same holds for $\varphi_{x=1}$. Thus the overall probability that D errs on the resulting formula is at least

$$p(u - \varepsilon) + q(1 - v - \varepsilon) + s(1 - w - \varepsilon) \geq 1/3 - \varepsilon.$$

In the case when one of the formulas $\varphi_{x=0}$, $\varphi_{x=1}$ is satisfiable we need that the second and the third numbers in (3) be at least 1/3.

This analysis is valid only for good r's. Take into account a fraction at most $n^{cd}2^{-n+2}$ of bad r's and also a fraction at most ε of r's such that S_r does not err on η_r. We obtain that the probability that D errs on the formula produced by $G(1^n)$ is at least to

$$1/3 - 2\varepsilon - n^{cd}2^{-n+2}.$$

It remains to show that there are nonnegative p, q, s such that $p+q+s = 1$ and all the numbers (3) are at least $1/3$. This happens for p, q, s such that all the three numbers (3) are equal. It is easy to see that this holds when p, q, s are proportional to $1/(2u-1), 1/(2v-1), 1/(2w-1)$. As all u, v, w are bounded away from $1/2$ (we are assuming that these numbers are at least $2/3$), all these numbers are bounded by a constant. Thus we are ably to find in polynomial time the desired p, q, s. For these p, q, s all the three numbers (3) are equal to their arithmetic mean, which equals to

$$\frac{1 + p(1-u) + q(1-v) + s(1-w)}{3} \geq \frac{1}{3}.$$

□

Remark 4. Theorem 5 remains true for $\varepsilon = 1/n^k$ for any constant k.

Remark 5. Say that NP $\neq$ RP *everywhere* if there is a constant c such that for every randomized SAT solver S and all $n > 1$, S errs on a formula of length between n and n^c. (A randomized SAT solver S errs on a formula φ if $S(\varphi) =$"don't know"with probability more $1/2$.)

If instead of NP $\neq$ RP we assume that NP $\neq$ RP everywhere then all our results hold in a stronger form: the quantifier "for infinitely many n" may be replaced by the universal quantifier.

Theorem 6. *If NP $\neq$ RP everywhere then (1) for every probabilistic SAT solver S and for every positive ε there is a sampler G such that* for all *n the algorithm S errs on the formula produced by $G(1^n)$ with probability at least $1-\varepsilon$, and (2) for every probabilistic polynomial time decision algorithm D and every positive ε there is a sampler G such that* for all *n the decision algorithm D errs on $G(1^n)$ with probability at least $1/3 - \varepsilon$.*

The proofs of the items of this theorem is entirely similar to those of Theorems 3 and 5 and thus we omit them. We only have to replace, in the definition of the search problem P, the requirement "the length of ψ is n" by the requirement "the length of ψ is between n and n^c" (and make a similar change in the definition of problem P'). The constructed sampler will work for almost all n, which is enough, as we can change its behaviour for the remaining n.

References

[1] L. M. Adleman. Two Theorems on Random Polynomial Time. FOCS 1978: 75-83

[2] Andrej Bogdanov and Luca Trevisan, Average-Case Complexity, Foundations and Trends in Theoretical Computer Science 1(2), 2006: 1–106.

[3] D. Gutfreund, R. Shaltiel, A. Ta-Shma If NP Languages are Hard on the Worst-Case, Then it is Easy to Find Their Hard Instances. Computational Complexity (CC) 16(4):412-441 (2007)

[4] Leonid A. Levin, Average Case Complete Problems. SIAM J. Comput. 15(1): 285–286 (1986)

ECCC ISSN 1433-8092

http://eccc.hpi-web.de