ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Reduction for NP-search Problems from Samplable to Uniform Distributions: Hard Distribution Case

Akinori Kawachi and Osamu Watanabe

February 2008, C–254

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Reduction for NP-search Problems from Samplable to Uniform Distributions: Hard Distribution Case

Akinori Kawachi and Osamu Watanabe
Graduate School of Information Science and Engineering,
Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo 152-8552, Japan
{kawachi,watanabe}@is.titech.ac.jp

Abstract

Impagliazzo and Levin showed a reduction from average-case hardness of any NP-search problem under any polynomial-time samplable distribution to that of another NP-search problem under the uniform distribution in [12]. Their target was the hardness of positive instances occurring with probability $1/\mathrm{poly}(n)$ under the distributions. In this paper, we focus on hardness of a larger fraction of instances. We reduce the hardness of positive instances for any NP-search problem occurring with probability $1-1/\mathrm{poly}(n)$ under any polynomial-time samplable distribution over positive instances to that for another NP-search problem with similar hardness under the uniform distribution. In order to illustrate the usage/importance of this technique, we show a simple way to modify the technique of Gutfreund, Shaltiel and Ta-Shma in [8] to construct an NP-search problem hard on average under the uniform distribution based on the assumption that NP $\neq$ RP and some worst-case mild derandomization holds.

1 Introduction

Reduction from samplable to uniform distributions. The theory of the average-case complexity has been studied extensively since 1970's to reveal the nature of "hard" problems in practical situations. In particular, many researchers recently discusses the average-case hardness of NP problems and relations to fundamental cryptographic primitives such as the one-way function from a cryptographic point of view (see, e.g., [1, 3, 16]) because cryptographic protocols need to withstand attacks from adversaries even if their instances (messages, keys, etc.) are chosen randomly and so we reduce the security of such cryptographic protocols to some NP problems believed to be hard on average in modern cryptography.

One of the seminal results in the average-case complexity theory is a completeness theorem on distributional NP problems under polynomial-time samplable distributions by Impagliazzo and Levin [12]. They showed that if we have some polynomial-time samplable distribution hard on average for some NP-search problem then we have another NP-search problem hard on average even under the uniform distribution.

Then the Impagliazzo-Levin reduction is stated as follows. For any polynomial-time samplable distribution schema $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$, NP-search problem F with support L (a set of positive instances of F), and $0 < \delta(n) < 1$, there exists an NP-search problem F' such that if we have some polynomial-time randomized search algorithm A' for F' with support L' satisfying

$$\Pr_{x:\mathcal{U}_n}[\, A' \text{ solves } x \mid x \in L' \,] > 1 - \delta(n),$$

then we have another polynomial-time randomized search algorithm A for F satisfying

$$\Pr_{x:\mathcal{H}_n}[\, A \text{ solves } x \mid x \in L \,] > 1 - \mathrm{poly}(n)\delta(n),$$

where $\mathcal{U}_n$ is the uniform distribution over instances of length n.

This result implies that the NP hardness under any polynomial-time samplable distribution is essentially equivalent to that under the uniform distribution within randomized polynomial time. An more explicit form of Impagliazzo-Levin reduction is given in Lemma 1 in [9]. Also, a survey on the average-case complexity theory by Bogdanov and Trevisan [2] contains the details of the proof.

In applications of the average-case complexity theory, we sometimes require higher hardness, e.g., hardness of all instances but a small fraction for an NP-search problem. A typical example is security reductions for cryptographic protocols and primitives. For example, if we want to show that some protocol is secure based on the one-way function, we prove that the security over all random instances but a negligible fraction is guaranteed from the hardness of inverting the one-way function over all inputs but a negligible fraction.

Unfortunately, one can see from the parameter of the success probability in the Impagliazzo-Levin reduction that their original result cannot treat such a large fraction of harder problem. It is therefore significant to develop new techniques for discussing harder problems in the Impagliazzo-Levin type reduction in some situation.

This issue is deeply related to the hardness amplification techniques since we could overcome the above barrier if the techniques would work well for the Impagliazzo-Levin reduction. Direct product theorems such as Yao's XOR lemma [19], which has many extensions and different proofs [14, 5, 11, 13, 15, 10, 17, 18], are very powerful techniques for hardness amplification. However, these techniques have an obvious disadvantage in a setting of the Impagliazzo-Levin reduction. They basically make an instance of a harder problem by taking direct product of multiple instances of the original NP-search problem. Thus, a fraction of positive instances exponentially decreases in the harder problem as we take the direct product over more instances if we want to keep the new problem in NP.

In this paper, we give a new reduction for an NP-search problem to directly treat harder distributions with the Impagliazzo-Levin type argument. The following informally describes our main result:

Theorem 1.1 (Informal) Let F be any NP-search problem with support L, and let be $\mathcal{H}$ be any polynomial-time samplable distribution schema $\mathcal{H}$ such that $\Pr_{x:\mathcal{H}_n}[\, x \in L \,] > n^{-\Omega(1)}$. Then, there exists another NP-search problem F' with support L' that satisfies:

1. We have for the new search problem F'

$$\Pr_{\bar{z}:\mathcal{U}_{\bar{n}}}[\, \bar{z} \text{ is a positive instance for } F' \text{ (i.e., } \bar{z} \in L') \,] > (\bar{n})^{-\Omega(1)}.$$

2. If some $(\bar{n})^a$-time randomized search algorithm A' for F' achieves

$$\Pr_{\bar{z}:\mathcal{U}_{\bar{n}}}[\, A' \text{ solves } \bar{z} \mid \bar{z} \in L' \,] > (\bar{n})^{-b},$$

some $n^{O(a)}$-time randomized search algorithm A achieves

$$\Pr_{x:\mathcal{H}_n}[\, A \text{ solves } x \mid x \in L \,] > n^{-\Omega(b)}.$$

The basic strategy of our proof is based on the same technique as the original one by Impagliazzo and Levin, the so-called hashing argument, which is one of strong and important tools broadly used in the computational complexity theory (e.g., [6]). A major difference is that we exploit a higher-performance hash functions than ones used in [12] due to the large fraction of hard instances.

Averagely hard NP-search problem. In order to illustrate the usage/importance of our Impagliazzo-Levin type reduction, we show a simple way to modify the technique of Gutfreund, Shaltiel and Ta-Shma in [8] to construct an NP-search problem hard on average under the uniform distribution based on assumptions that NP $\neq$ RP and worst-case mild derandomization hold.

One of challenging problems in the average-case complexity theory is construction for NP problems hard on average or the one-way function from a worst-case NP-hardness assumption such as NP $\neq$ RP. However, there have been actually several negative results for such worst-case to average-case reductions on the NP-hardness so far [4, 3, 1, 16]. These results show that if we have such reductions (e.g., a non-adaptive reduction from inverting one-way functions to NP-hard problems) then something strange happens in the context of the computational complexity theory (e.g., co$NP \subseteq$ AM).

Here, it should be noticed that they assume the *black-box* reductions. It would be thus possible to avoid these negative evidences if we exploit a *non-black-box* technique for constructing the reductions, as noted in several of these papers.

Gutfreund, Shaltiel, and Ta-Shma [8] actually developed an innovative technique for proving a worst-case to average-case reduction that is not fully black-box. By this technique, they obtained a superpolynomial-time samplable distribution such that no efficient randomized algorithm can correctly solve SAT instances occurring some constant probability under the distribution. Going along the same line, Gutfreund [7] gave results of separating some average-case complexity classes by constructing an NP-search problem highly hard on average under some samplable distribution from a worst-case hardness assumption.

Most recently, the power of the technique was explicitly demonstrated by Gutfreund and Ta-Shma [9]. They constructed a quasi-NP decision problem (that has polynomial witness length and superpolynomial verification time.) hard on average under the uniform distribution from the assumption that NP $\neq$ RP and some average-case mild derandomization holds. They also gave a negative evidence on the same result in the case of the black-box reduction.

In this paper, we provide a search version of [9] for an NP-search problem hard on a large fraction over positive instances under the uniform distribution. Informally, we prove the following theorem:

Theorem 1.2 (Informal) Assume that NP $\neq$ RP hold. For any constant $a \geq 1$ and $b > 0$, we have an NP-search problem F with support L that satisfies the following:

1. L is large enough under the uniform distribution; namely, for all sufficiently large n, we have
$$\Pr_{x:\mathcal{U}_n}[x \in L] > n^{-c}$$
for some constant c independent of a and b.
2. Any $O(n^a)$-time randomized search algorithm fails to yield a solution to F badly; namely, we have
$$\Pr_{x:\mathcal{U}_n}\left[\Pr_A[A \text{ solves } x] \leq n^{-b} \,\middle|\, x \in L\right] > 1 - n^{-\Omega(1)}.$$

The derandomization assumption will be formally defined in Assumption 3.1, which is stronger than the average-case one in [9].

In Gutfreund and Ta-Shma's construction for the quasi-NP problem, they applied the original Impagliazzo-Levin reduction to some distribution and then amplified the hardness using the NP-hardness amplification technique given by Trevisan [18]. Those techniques make the problem hard against any decision algorithm with a polylogarithmic advantage (i.e., $1/2 + \text{poly}\log n$). On the other hand, our target is a search problem hard against any search algorithm with polynomially small success probability. It is then essential to apply our reduction to this case instead of the original one for treating the small probability.

2 Reduction from Samplable to Uniform Distributions

First, we introduce some notion and notations used below. $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ and $\mathcal{U} = \{\mathcal{U}_n\}_{n\in\mathbb{N}}$ denotes the hard distribution we consider and the uniform distribution schemata, where $\mathcal{H}_n$ and $\mathcal{U}_n$ are the distributions over $\{0,1\}^n$. We use notations "$\mathcal{H}_n(x)$" and "$x : \mathcal{H}_n$" to represent a probability of occurring an instance x and a random instance x occurred according to a distribution $\mathcal{H}_n$ respectively. By $H(1^n)$, we denote a random variable representing an output from a sampler H for $\mathcal{H}$ on a length parameter 1^n. If we argue random strings in the algorithm, we denote it by $H(1^n; r)$.

We say that F is an NP search problem with support L if we can verify the correctness of its solutions in a set L of positive instances in polynomial time. Here we say the "solution" as a witness for a positive instance. Note that the answer "NO" for a negative instance is not a solution. We also describe $F(x)$ as a set of correct solutions to a positive instance x for a search problem F. For simplification, we assume that all solutions to inputs of length n has the same length n'.

The following is the formal statement of Theorem 1.1.

Theorem 2.1 We have some constants c_0 and e_0 with which the following statement holds. Let F be any NP-search problem with support L, and let $\mathcal{H}$ and H be any polynomial-time samplable distribution for F and its sampler. Suppose that F, $\mathcal{H}$, and H satisfy the following for some constants d and s:

1. $\Pr_{x:\mathcal{H}_n}[\, x \in L \,] \geq n^{-d}$,
2. $H(1^n)$ generates an instance of size n according to the distribution $\mathcal{H}_n$, and
3. $H(1^n)$ requires a random seed of size n^s and runs in $n^{O(1)}$ time.

Then there exists a constant d_0 such that we have some NP-search problem F' with support L' that satisfies the following:

1. $\Pr_{\bar{z}:\mathcal{U}(\overline{n})}[\, \bar{z} \in L' \,] \geq (\overline{n})^{-(d_0+d)}$,
2. if some $(\overline{n})^a$-time randomized search algorithm A' for F' achieves

$$\Pr_{\bar{z}:\mathcal{U}_{\overline{n}}}\Big[\ \Pr_{A'}[\, A' \text{ solves } \bar{z} \,] \ \geq\ 1 - n^{-O(1)} \mid \bar{z} \in L' \ \Big] \ \geq\ (\overline{n})^{-b}, \tag{1}$$

   then by using A' some $n^{c_0\cdot s\cdot a}$-time randomized search algorithm A solves F, i.e.,

$$\Pr_{x:\mathcal{H}(n)}\Big[\ \Pr_{A}[\, A \text{ solves } x \,] \ \geq\ n^{-e_0\cdot s\cdot b} \mid x \in L \ \Big] \ \geq\ n^{-e_0\cdot s\cdot b}. \tag{2}$$

Remark 1. In the theorem, we omit some details for the sake of simplifying statements. Some unimportant constants and polynomials are not explicitly stated. Though not explicitly stated, size parameters n and $\overline{n}$ are any sufficiently large numbers and other constants as well as algorithms are independent of these size parameters.

The rest of this section is devoted for proving this theorem. For the proof, we need hash function families satisfying certain properties. More specifically, we use highly independent hash function families stated in the following two propositions. (The proofs are standard and they are omitted here.)

Proposition 2.2 For any n and k such that $k \leq n$, there exists a hash function family $\mathrm{Hash}_3(n,k)$ mapping $\{0,1\}^n$ to $\{0,1\}^k$ satisfying the following: Any $h \in \mathrm{Hash}(n,k)$ is of polynomial length and computable in polynomial time, For any distinct t elements x_1, x_2, x_3 of $\{0,1\}^n$, and for any elements y_1, y_2, y_3 of $\{0,1\}^k$, we have

$$\Pr_{g:\mathrm{Hash}_3(n,k)}[\, g(x_1) = y_1 \ \wedge\ g(x_2) = y_2 \wedge\ g(x_3) = y_3 \,] = (2^{-k})^3,$$

where "$g : \mathrm{Hash}_3(n,k)$" means that a hash function g is sampled from $\mathrm{Hash}_3(n,k)$ uniformly at random.

Proposition 2.3 Let n and k be any positive integers such that $k \leq n$. There exists a hash function family $\mathrm{Hash}(n,k)$ satisfying the following: Any $h \in \mathrm{Hash}(n,k)$ is of polynomial length and computable in polynomial time, and for any

$e > 0$ and any $X \subseteq \{0,1\}^n$ of size $\geq n^{e+f_0} \cdot 2^k$, we have

$$\Pr_{h:\mathrm{Hash}(n,k)}\left[\ |h(X)| \geq (1-n^{-e})2^k\ \right] \ \geq\ 1-n^{-e},$$

where "$h : \mathrm{Hash}(n,k)$" means that a hash function h is sampled from $\mathrm{Hash}(n,k)$ uniformly at random.

Now we start our proof by defining our target problem F' based on a given search problem F and a sampler H. Our approach is almost the same as Impagliazzo and Levin's [12]; only the difference is to use the highly independent hash function family.

As already explained, we define our search problem by specifying the following set $F'(\bar{z})$ of valid solutions to a given input $\bar{z} = \langle k, h, y, g, pad\rangle$.

$$\begin{aligned} F'(\bar{z}) \quad := \quad \{\ \langle x, r_1, r_2, r_3, w\rangle \ \mid \\ &\text{(a) } h(x) = y, \\ &\text{(b) } w \in F(x) \text{ (i.e., } w \text{ is one of the solution for } F), \\ &\text{(c) } H(n; r_1) = H(n; r_2) = H(n; r_3) = x, \text{ and} \\ &\text{(d) } g(r_1) = 000\cdots0 \wedge g(r_2) = 010\cdots0 \wedge g(r_3) = 100\cdots0\ \ \}. \end{aligned}$$

We clarify the domain of input and output components w.r.t. the size parameter n ($= |x|$). First we fix some more size parameters. Following our notation rule, we use n' to denote the length of solution for x; we may assume that $n' \leq n$. Let m be the length of random seeds used by $H(1^n)$; by the condition of the theorem, we have $m = n^s$. Symbols h and g in the input $\bar{z}$ denote binary descriptions of hash functions in $\mathrm{Hash}(n,k)$ and $\mathrm{Hash}_3(m, m-(k+2\log n))$ respectively; we may assume that their length are polynomially bounded by n and n^s. Thus, with a padding pad of appropriate length, we assume that $\bar{n} = |\bar{z}| = n^{s\cdot l_0}$ for some constant l_0.

An input $\bar{z} = \langle k, h, y, g, pad\rangle$ consists of five components, where their domains are: (i) k is from $[n']$, (ii) h is from $\mathrm{Hash}(n,k)$, (iii) y is from $\{0,1\}^k$, (iv) g is from $\mathrm{Hash}_3(m, m-(k+2\log n))$, and (v) pad is from $\{0,1\}^l$ for some appropriate length l. On the other hand, the domain of components of output $\langle x, r_1, r_2, r_3, w\rangle$ are: (i) x is from $\{0,1\}^n$, (ii) r_1, r_2, r_3 are from $\{0,1\}^m$, and (iii) w is from $\{0,1\}^{n'}$. Below we often write $\bar{r}$ for (r_1, r_2, r_3).

By the uniform distribution $\mathcal{U}_{\bar{n}}$ for $\bar{z}$, we mean that k, h, y, g, and pad are chosen uniformly at random* from their domains.

Before proving that this F' indeed satisfies the theorem, let us recall the intuitive idea of [12] for introducing this problem.

We would like to convert the distribution $\mathcal{H}_n$ to a uniform distribution. For this purpose, we encode each $x \in \{0,1\}^n$ by a string y of length reflecting its *weight* $\mathcal{H}_n(x)$. For example, a string x_1 with $\mathcal{H}_n(x_1) = 2^{-2}$ is encoded by some y_1 of roughly 2 bit length whereas a string x_2 with $\mathcal{H}_n(x_1) = 2^{-10}$ is encoded by some y_2 of roughly 10 bit length. Note that there are at most 4 strings with heavy weight 2^{-2}; hence, 2 bit string is enough. On the other hand, we may need 10 bits to encode strings with light weight 2^{-10}. Codes y_1 and y_2 are obtained by random hash functions. That is, by using randomly chosen $h_1 \in \mathrm{Hash}_2(n,2)$ and $h_2 \in \mathrm{Hash}_2(n,10)$, we compute $y_1 = h_1(x_1)$ and $y_2 = h_2(x_2)$. Then we may regard $\langle h_1, y_1\rangle$ and $\langle h_2, y_2\rangle$ as uniformly generated random strings. Furthermore, with random padding pad_1 and pad_2 of appropriate length, two strings $\langle h_1, y_1, pad_1\rangle$ and $\langle h_2, y_2, pad_2\rangle$ can be regarded as random strings of the same length.

This encoding has the following problem: by using, e.g., h_1, light inputs such as x_2 are also mapped to short codes such as y_1. Then we cannot guarantee that some useful information on x_1 is obtained by solving the instance $\langle h_1, y_1, pad_1\rangle$. Hash function g is used to avoid this problem; it is used to check whether a string x mapped to y_1 has weight $\mathcal{H}_n(x)$ large

*Precisely speaking, these components are obtained from one random binary string of certain length. In particular, we need to specify a way to split a binary string into five components and a way to generate hash functions h and g. For generating hash functions, we use some samplers that are implicitly assumed for hash function families; that is, h and g are generated by these samplers by using a part of the random binary string as random seeds. The details are omitted.

enough. Note that weight $\mathcal{H}_n(x)$ is essentially the same as the number of random seeds r with which the generator $H(1^n; r)$ yields x. The hash function g and the condition (d) for solutions of F' are used to check whether there are enough number of such random seeds for x.

Now let us prove that F' satisfies the theorem. Below size parameters are fixed as above and variable symbols are also defined as above. We use $X \subseteq \{0,1\}^n$ to denote the set of all instances of length n that are positive for F.

Remark 2. For avoiding unnecessary complications in our arguments, we introduce the following two assumptions. First we assume that $\mathcal{H}_n(X) = 1$; that is, our sampler generates only positive instances for F. We can easily modify our probability analysis for the general case. Secondly we assume that the algorithm A assumed in the theorem is deterministic. Again the modification for the general situation is easy; we introduce one more variable for a random seed used by A and consider the execution of the algorithm under several fixed "good" random seeds. The details are omitted here.

We first give the statement 1 of Theorem 2.1. That is, the following lemma.

Lemma 2.4 For some constant $d_0 > 0$, the following holds:

$$\Pr_{\overline{z}:\mathcal{U}_{\overline{n}}}[\ F'(\overline{z}) \neq \emptyset\ ] \ \geq\ (\overline{n})^{-d_0}.$$

Proof. For each i, $0 \leq i \leq m$, define X_i by

$$X_i \ =\ \{\, x \,|\, 2^{-(i+1)} < \mathcal{H}_n(x) \leq 2^{-i}\,\}.$$

Noting that $m = n^s$ and $|X| \leq 2^n$, we have (for sufficiently large n) that

$$\sum_{i \geq n+s\log n+1} \mathcal{H}_n(X_i) \ \leq\ m \cdot \frac{2^n}{2^{n+s\log n+1}} \ \leq\ \frac{1}{2}.$$

Thus, there is some i_0, $0 \leq i_0 \leq n + s\log n$, such that $\mathcal{H}_n(X_{i_0}) \geq 1/2(n + s\log n + 1) > 1/4n$. Let $k_0 = \min(i_0, n)$, and for bounding the probability $\Pr_{\overline{z}:\mathcal{U}(\overline{n})}[\ F'(\overline{z}) \neq \emptyset\ ]$, we consider the probability that $\langle k_0, h, y, g, pad\rangle$ has a solution. Here we note that $|X_{i_0}| \geq 2^{i_0}/4n$ because $\mathcal{H}_n(x) \leq 2^{-i_0}$ for all $x \in X_{i_0}$ and $\mathcal{H}_n(X_{i_0}) \geq 1/4n$. On the other hand, since X_{i_0} is a subset of $\{0,1\}^n$, we have $|X_{i_0}| \leq 2^n$. Hence, from

$$\frac{2^{i_0}}{4n} \ \leq\ |X_{i_0}| \ \leq\ 2^n,$$

it follows that $i_0 \leq n + 2\log n$. Thus, we have $k_0 \leq i_0 \leq k_0 + 2\log n$.

Let $H^{-1}(1^n, x) = \{r | H(1^n; r) = x\}$; then we have $|H^{-1}(1^n, x)| \geq 2^{m-(i_0+1)}$ for all $x \in X_{i_0}$. Now by using this and the bound $|X_{i_0}| \geq 2^{i_0}/4n$ derived above, and also by using the independence properties of h and g, we have the following

bound.

$$
\begin{aligned}
&\Pr_{h,y,g,pad}[\ F'(\langle k_0, h, y, g, pad\rangle) \neq \emptyset\ ] \\
&\geq \Pr_{h,y,g,pad}[\ \exists x, \bar{r}, w\,[\ \text{(a)}\sim\text{(d) holds for } h, y, g, x, \bar{r}, w\ ] \\
&\geq \sum_{x \in X_{i_0}} \Pr_{h,y,g,pad}[\ \exists \bar{r}\,[\ \text{(a),(c),(d) holds}^{*1} \text{ for } h, y, g, \bar{r} \text{ on } x\ ] \\
&\qquad - \sum_{x \neq x' \in X_{i_0}} \Pr_{h,y,g,pad}[\ (\cdots \text{ on } x) \wedge (\cdots \text{ on } x')\ ] \\
&\qquad\qquad \text{(Note } *1\text{: (b) is satisfied by considering only } x \in X_{i_0}) \\
&\geq \frac{1}{2} \cdot \sum_{x \in X_{i_0}} \Pr_{h,y}[\ \text{(a) for } h, y, x\ ] \cdot \Pr_{g}[\ \exists \bar{r}\,[\ \text{(c),(d) for } g, \bar{r}, x\ ] \\
&\geq \frac{|X_{i_0}|}{2} \cdot \sum_{y \in \{0,1\}^{k_0}} \frac{1}{2^{k_0}} \cdot \Pr_{h}[\ \text{(a) for } h, y, x\ ] \cdot \sum_{\bar{r}:*2} \Pr_{g}[\ \text{(d) for } g, \bar{r}\ ] \\
&\text{(Note } *2\text{: } \bar{r} = (r_1, r_2, r_3) \text{ are three different elements of } H^{-1}(1^n, x)) \\
&\geq \frac{|X_{i_0}|}{2} \cdot \frac{2^{k_0}}{2^{k_0} \cdot 2^{k_0}} \cdot \frac{2^{m-(i_0+1)} \cdot (2^{m-(i_0+1)} - 1) \cdot (2^{m-(i_0+1)} - 2)}{2^{m-(k_0+2\log n)} \cdot 2^{m-(k_0+2\log n)} \cdot 2^{m-(k_0+2\log n)}} \\
&\geq \frac{2^{i_0}}{8n} \cdot \frac{2^{k_0}}{2^{k_0} \cdot 2^{k_0}} \cdot \frac{(2^{m-(i_0+1)})^3}{2 \cdot (2^{m-(k_0+2\log n)})^3} \\
&\geq \frac{1}{8n} \cdot \frac{1}{16} \geq \frac{1}{128n}
\end{aligned}
$$

Since this is a bound for the case $k = k_0$, by considering the probability that $k = k_0$, we have $\Pr_{\bar{z}:\mathcal{U}(\bar{n})}[\ F'(\bar{z}) \neq \emptyset\ ] \geq 1/(128n(n+1))$. We may assume that $\bar{n} > 128(n+1)$; therefore, the desired bound is shown with $d_0 = 2$. □

Next consider the second statement of the theorem. Here we assume some $(\bar{n})^a$-time algorithm A' satisfying the inequality (1) of Theorem 2.1 for all sufficiently large $\bar{n}$. Now for our algorithm A (for solving F), we consider the following simple one.

Algorithm: A (input x)

1. Choose $k \in [n']$, $h \in \mathrm{Hash}(n, k)$, $g \in \mathrm{Hash}_3(m, m-(k+2\log n))$, and pad uniformly at random.
2. Run $A'(\langle k, h, h(x), g, pad\rangle)$ and output the last component w of the obtained output.

Clearly, the running time of this algorithm is $O(\bar{n}^c + \bar{n}^a)$ for some constant $c > 0$, and this can be bounded by $n^{c_0 \cdot s \cdot a}$ with some constant c_0 because $\bar{n} = n^{s \cdot l_0}$.

In the rest of this section, we prove that this A achieves the desired performance stated as (2). That is, we show the following lemma.

Lemma 2.5 For some constant $e_0 > 0$, the following holds:

$$
\Pr_{x:\mathcal{H}_n}[\ \Pr_{A}[\ A(x) \text{ yields some } w \in F(x)\ ] \geq n^{-e_0 \cdot s \cdot b}\ \mid\ x \in L\ ] \geq n^{-e_0 \cdot s \cdot b}.
$$

We prove the lemma by a sequence of claims. First we analyze the performance of the algorithm A'. From the condition (1), we have the following. (Recall that we assumed for simplicity that A is deterministic.)

$$
\Pr_{\bar{z}=\langle k,h,y,g,pad\rangle}[\ A'(\bar{z}) \in F'(\bar{z})\,|\,\bar{z} \in L\ ] \geq (\bar{n})^{-b} = n^{-sl_0 b}.
$$

Below we drop the above condition "... | $\bar{z} \in L$" of the probability for notational simplification. We implicitly consider this conditional sample space when the random variables contain components of an instance of F'.

Thus, we have the following bound for some $k_1 \in [n']$. (Below we use $\bar{z}_1$ or sometimes $\bar{z}_1\langle h, y\rangle$ as a shorthand of $\langle k_1, h, y, g, pad\rangle$.)

$$\Pr_{h,y,g,pad}[\, A(\bar{z}_1) \in F'(\bar{z}_1)\,] \;\geq\; n^{-s\cdot l_0\cdot b}/(n+1) \;\geq\; n^{-(s\cdot l_0\cdot b+2)}.$$

Here we note the following variation of the Markov inequality, which will be used in the following argument.

Proposition 2.6 Consider any index set X and a set of values $\{p_x\}_{x\in X}$ such that $0 \leq p_x \leq 1$ for all $x \in X$. Then we have

$$\frac{\sum_{x\in X} p_x}{|X|} \geq \gamma \;\Rightarrow\; \frac{|\{x \mid p_x \geq \gamma/2\}|}{|X|} \;\geq\; \frac{\gamma}{2}.$$

Let $\gamma = n^{-(sl_0b+2)}$ and apply this proposition twice to the above bound. Then we have the following claims.

Claim 1 We say that h is *good* if $\Pr_{y,g,pad}[A'(\bar{z}_1) \in F'(\bar{z}_1)] \geq \gamma/2$. The proportion of good h is at least $\gamma/2$. That is,

$$\Pr_{h}\left[\, \Pr_{y,g,pad}[\, A'(\bar{z}_1) \in F'(\bar{z}_1)\,] \;\geq\; \frac{\gamma}{2} \,\right] \;\geq\; \frac{\gamma}{2}.$$

Claim 2 Consider any good h and fixed. We say that y is *good* (w.r.t. h) if $\Pr_{g,pad}[A'(\bar{z}_1) \in F'(\bar{z}_1)] \geq \gamma/4$. The proportion of good y is at least $\gamma/4$. That is,

$$\Pr_{y}\left[\, \Pr_{g,pad}[\, A'(\bar{z}_1) \in F'(\bar{z}_1)\,] \;\geq\; \frac{\gamma}{4} \,\right] \;\geq\; \frac{\gamma}{4}.$$

At this point, we set some more parameters as follows:

$$\begin{aligned} e &:= 1 + (sl_0b + 2), \;\; \text{(so that } n^{-e} < \gamma/8\text{)} \\ f &:= f_0 + e, \;\; \text{(where } f_0 \text{ is from Proposition 2.3)} \\ K &:= 2^{k_1}, \;\; \text{and } |X_{\mathrm{fat}}| = n^f K. \;\; \text{(see below for the definition of } X_{\mathrm{fat}}\text{)} \end{aligned}$$

We say that $x \in X$ is *fat* if its weight $\mathcal{H}_n(x)$ satisfies $\mathcal{H}_n(x) \geq 1/(n^f K)$. Note that there are at most $n^f K$ fat instances in X. Let X_{fat} denote a set consisting of all fat instances in X and some dummy strings[†] so that $|X_{\mathrm{fat}}| = n^f K$. Then the following claim holds.

Claim 3

$$\begin{aligned} &\Pr_{h,y,g}[\, \exists x \in X_{\mathrm{fat}}[\, h(x) = y\,] \wedge A'(\bar{z}_1) \in F'(\bar{z}_1)\,] \\ &\qquad \geq\; \left(\frac{\gamma}{2} - \frac{\gamma}{8}\right)\left(\frac{\gamma}{4} - \frac{\gamma}{8}\right)\cdot\frac{\gamma}{4} \;\geq\; \frac{\gamma}{128} \;\geq\; \frac{1}{16n^e} \end{aligned}$$

Proof. The claim is proved by counting all h, y, and g satisfying the condition. For our X_{fat}, we say that h is *nonshrink* if $|h(X_{\mathrm{fat}})| \geq K(1 - n^{-e}) \geq K(1 - \gamma/8)$. By Proposition 2.3, the proportion of nonshrink h's is at least $1 - \gamma/8$. On the other hand, the proportion of good h's is at least $\gamma/2$. Hence, the probability that random h is both nonshrink and good h is $\geq \gamma/2 - \gamma/8$.

Similarly, for each nonshrink and good h, we have at least $K(1 - \gamma/8)$ y's that have some $x \in X_{\mathrm{fat}}$ such that $y = h(x)$. On the other hand, there are at least $K\gamma/4$ good y's, i.e., y's for which $A(\langle k_1, h, y, g, pad\rangle) \in F'(\bar{z}_1)$ holds for at least $\gamma/4$ of all g's and *pad*'s. Hence, the probability that random y, g, and *pad* satisfies both $y = h(x)$ and $A'(\bar{z}_1) \in F'(\bar{z}_1)$ is at least $(\gamma/4 - \gamma/8)\cdot\gamma/4$. Putting these bounds together, we have the bound of the claim. □

Let us further analyze the bound of the above claim. Here we divide the event $A'(\bar{z}_1) \in F'(\bar{z}_1)$ into *disjoint* subcases

[†] It may be the case that $n^f K \geq 2^n$. Then $X_{\mathrm{fat}} = \{0,1\}^n$; analysis for this case is easier and omitted.

by considering the output of A.

$$\begin{aligned}
\frac{1}{16n^e} &\leq \Pr_{h,y,g,pad}[\ \exists x \in X_{\mathrm{fat}}[\ h(x) = y\ ] \wedge A'(\bar{z}_1) \in F'(\bar{z}_1)\ ] \\
&= \sum_{x'} \Pr_{h,y,g,pad}[\ \exists x \in X_{\mathrm{fat}}[\ h(x) = y\ ] \wedge A'(\bar{z}_1) = (x', \bar{r}, w) \in F'(\bar{z}_1)\ ] \\
&= \sum_{x' \in X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ \exists x \in X_{\mathrm{fat}}[\ h(x) = y\ ] \wedge A'(\bar{z}_1) = (x', \bar{r}, w) \in F'(\bar{z}_1)\ ] \\
&\quad + \sum_{x'' \notin X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ \exists x \in X_{\mathrm{fat}}[\ h(x) = y\ ] \wedge A'(\bar{z}_1) = (x'', \bar{r}, w) \in F'(\bar{z}_1)\ ] \\
&\leq \sum_{x' \in X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ A'(\bar{z}_1) = (x', \bar{r}, w) \in F'(\bar{z}_1)\ ] \qquad (3) \\
&\quad + \sum_{x'' \notin X_{\mathrm{fat}}} \sum_{x \in X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ h(x) = y \wedge A'(\bar{z}_1) = (x'', \bar{r}, w) \in F'(\bar{z}_1)\ ] \qquad (4)
\end{aligned}$$

Consider the last two terms, i.e., (3) and (4). Noting that $h(x') = y$ is a part of the condition $(x', \bar{r}, w) \in F'(\bar{z}_1)$, we can restate (3) as follows.

$$\begin{aligned}
(3) &= \sum_{x' \in X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ h(x') = y \wedge A'(\bar{z}_1) = (x', \bar{r}, w) \in F'(\bar{z}_1)\ ] \\
&= \sum_{x' \in X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ A'(\bar{z}_1) = (x', \bar{r}, w) \in F'(\bar{z}_1) \mid h(x') = y\ ] \cdot \Pr_{h,y}[\ h(x') = y\ ] \\
&= \sum_{x' \in X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ A'(\bar{z}_1\langle h, y\rangle) = (x', \bar{r}, w) \in F'(\bar{z}_1\langle h, y\rangle) \mid h(x') = y\ ] \cdot \frac{1}{K} \\
&= \sum_{x' \in X_{\mathrm{fat}}} \frac{1}{K} \cdot \Pr_{h,y,g,pad}[\ A'(\bar{z}_1\langle h, h(x')\rangle) = (x', \bar{r}, w) \in F'(\bar{z}_1\langle h, h(x')\rangle)\ ] \\
&= \frac{1}{K} \cdot \sum_{x' \in X_{\mathrm{fat}}} \Pr_{h,g,pad}[\ A'(\bar{z}_1\langle h, h(x')\rangle) = (x', \bar{r}, w) \in F'(\bar{z}_1\langle h, h(x')\rangle)\ ]
\end{aligned}$$

Intuitively, this is the total success probability of our procedure A. On the other hand, the term (4) is the probability that A's answer does not help us for solving $x' \in X_{\mathrm{fat}}$. Our new technique of using highly independent hash functions and our new analysis, which is different from the one in [12], are for bounding this probability small. More specifically, we can bound as the following claim, which is the key of our argument.

Claim 4 For any $x \in X_{\mathrm{fat}}$, we have

$$\sum_{x'' \notin X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ h(x) = y \wedge A'(\bar{z}_1) = (x'', \bar{r}, w) \in F'(\bar{z}_1)\ ] \leq \frac{n^6}{n^f \cdot n^f K}.$$

Thus, by if $f_0 \geq 7$, we have

$$(4) \leq \sum_{x \in X_{\mathrm{fat}}} \frac{n^6}{n^f \cdot n^f K} = \frac{n^6}{n^f} = \frac{n^6}{n^{f_0+e}} \leq \frac{1}{32n^e}.$$

Proof. The claim is proved by the following analysis.

$$\begin{aligned}
&\sum_{x'' \notin X_{\mathrm{fat}}} \Pr_{h,y,g,pad}[\ h(x) = y \wedge A'(\bar{z}_1) = (x'', \bar{r}, w) \in F'(\bar{z}_1)\ ] \\
&\leq \sum_{x'' \notin X_{\mathrm{fat}}} \sum_{r_1:*1} \sum_{r_2:*1} \sum_{r_3:*1} \Pr_{h,y,g,pad}[\ h(x) = h(x'') = y \wedge \text{(d) for } g \text{ and } \bar{r}\ ] \\
&\qquad\qquad \text{(Note } *1\text{: } r_i \text{ is chosen so that } H(1^n; r_i) = x'') \\
&\leq \sum_{r_1:*2} \sum_{r_2:*3} \sum_{r_3:*3} \Pr_{h,y,g,pad}[\ h(x) = h(z) = y \wedge \text{(d) for } g \text{ and } \bar{r}\ ] \\
&\qquad\qquad \text{(Note } *2\text{: } r_1 \text{ is chosen so that } H(1^n; r_1) \notin X_{\mathrm{fat}}) \\
&\qquad \text{(Note } *3\text{: letting } z = H(1^n; r_1), \ r_i \text{ is chosen so that } H(1^n; r_i) = z) \\
&\leq 2^m \cdot \frac{2^{m-k_1}}{n^f} \cdot \frac{2^{m-k_1}}{n^f} \cdot \frac{1}{K^2} \cdot \left(\frac{1}{2^{m-(k_1+2\log n)}} \right)^3 = \frac{n^6}{Kn^f \cdot n^f}.
\end{aligned}$$

Here we use the fact that $z \notin X_{\mathrm{fat}}$ implies that z is not fat and $\mathcal{H}_n(z) < 1/(n^f K)$; in other words, the number of r such that $H(1^n; r) = z$ is less than $2^{m-k_1}/n^f$. □

From the above claim and the restatement of (3), we have

$$\frac{K}{32n^e} \leq \sum_{x' \in X_{\mathrm{fat}}} \Pr_{h,g,pad}[\ A'(\bar{z}_1\langle h, h(x')\rangle) = (x', \bar{r}, w) \in F'(\bar{z}_1\langle h, h(x')\rangle)\ ].$$

From this we now show that there are enough x''s for which A succeeds with our desired probability.

We say that x is *good* if $\Pr_{h,g,pad}[A'(\bar{z}_1\langle h, h(x)\rangle) = (x, \bar{r}, w) \in F'(\bar{z}_1)] \geq 1/(64n^{e+f})$. We choose the constant e_0 of the lemma large enough so that $1/(64n^{e+f}) \geq n^{-e_0 \cdot sb}$; hence, good x are those for which $A(x)$ has the desired success probability. Therefore, the lemma is proved by the following claim.

Claim 5

$$\Pr_{x:\mathcal{H}_n}[\ x \text{ is good}\ ] \geq 1/64n^{e+f}.$$

Proof. First we apply Proposition 2.6 to the above bound. Note $|X_{\mathrm{fat}}| = n^f K$; hence, we have

$$\left| \left\{ x \;\middle|\; \Pr_{h,g,pad}[A'(\bar{z}_1\langle h, h(x)\rangle) = (x, \bar{r}, w) \in F'(\bar{z}_1)] \geq \frac{1}{64n^{e+f}} \right\} \right| \geq \frac{|X_{\mathrm{fat}}|}{64n^{e+f}}.$$

Next we show that a good x is in fact fat; that is, for any good x, we have

$$\Pr_r[\ H(1^n; r) = x\ ] \geq \frac{1}{n^f K} = \frac{2^{m-k_1}}{n^f \cdot 2^m}.$$

This is because if otherwise, we have

$$\Pr_g[\ \exists \bar{r}\ [\ H(1^n; r_1) = H(1^n; r_2) = H(1^n; r_3) = x \wedge \text{(d) holds for } \bar{r} \text{ and } g\ ]\ ] < n^6/n^{3f},$$

but then the probability that random g has some $\bar{r}$ satisfying the condition (d) for the solution is much smaller than $1/(64n^{e+f})$; hence, $A(x)$'s success probability becomes so small that x cannot be good.

Now we know that there are at least $|X_{\mathrm{fat}}|/(64n^{e+f})$ $(= K/(64n^e))$ good x's and that each of them is fat, i.e., $\mathcal{H}_n(x) \geq 1/(n^f K)$. This proves that the probability of good x under the distribution $\mathcal{H}_n$ is at least $1/(64n^{e+f})$. □

3 Averagely Hard NP-search Problem

We next apply our new technique to a worst-case to average-case reduction for NP-search problems. As mentioned in Theorem 1.2 of Section 1 informally, we construct an NP-search problem hard to yield a solution on average under the uniform distribution.

Before giving the formal statement, we explicitly describe the mild derandomization assumption appeared in Theorem 1.2.

Assumption 3.1 Let B_1 and B_2 be two randomized decision algorithms. We say that B_1 and B_2 are δ-indistinguishable if

$$\left|\Pr_{B_1}[B_1(x) = \text{“YES”}] - \Pr_{B_2}[B_2(x) = \text{“YES”}]\right| \leq \delta$$

for any $x \in \{0,1\}^*$. For any polynomial-time randomized decision algorithm B and any constant $\varepsilon > 0$, there exists a probabilistic polynomial-time decision algorithm B^ε such that B^ε requires a random seed of size at most n^ε on input length n, and B^ε and B are $1/100$-indistinguishable.

This assumption is stronger than the original one used in [9]. The original one is that for any samplable distribution we have some derandomized algorithm that works well over instances occurring with a constant probability, say $99/100$, (i.e., average-case derandomization), but we now assume that a derandomized algorithm works well over any instance (i.e., worst-case derandomization). Similarly to the case of [9], note that our assumption does not directly lead to a strong consequence such as BPP = P or collapse of the BPTIME hierarchy.

Combining Assumption 3.1 with our reduction given in Section 2, we obtain the following theorem.

Theorem 3.2 Assume that NP $\neq$ RP and Assumption 3.1 holds. For any constant $a \geq 1$ and $b > 0$, we have an NP-search problem F' with support L' that satisfies the following:

1. L' is large enough under the uniform distribution; namely, for all sufficiently large n, we have

$$\Pr_{x:\mathcal{U}_n}[x \in L'] > n^{-c}$$

   for some constant c independent of a and b.

2. Any n^a-time randomized search algorithm fails to yield a solution to F' badly; namely, we have

$$\Pr_{x:\mathcal{U}_n}\left[\Pr_A[A' \text{ solves } x] \leq n^{-b} \,\middle|\, x \in L'\right] > 1 - 2^{-n^{1/d}},$$

   where d is a constant depending only on a and b.

We omit the proof of this theorem since it can be easily obtain by combining our reduction of Theorem 2.1 with hard instance sampler stated in Theorem 3.3 below.

Theorem 3.3 Let $a > 1$ and $b > 0$ be any fixed constants and let $d > 0$ be some constant depending only on a and b. If Assumption 3.1 holds and NP $\neq$ RP, there exists a samplable distribution schema $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ such that for any n^a-time randomized search algorithm A (assignment finding problem), infinitely many $n \in \mathbb{N}$

$$\Pr_{x:\mathcal{H}_n}\left[\Pr_A[A \text{ solves } x] \leq n^{-b} \,\middle|\, x \in \text{SAT}\right] > 1 - 2^{-n^{1/d}} \quad \text{and} \quad \Pr_{x:\mathcal{H}_n}[x \in \text{SAT}] > 3/8.$$

Then, the sampler H for $\mathcal{H}$ runs in polynomial time and requires a random seed of size $O(n^5)$.

As observed in [9], the instance size of Impagliazzo-Levin type reduction, including ours, depends on the size of random seeds to generate the samplable distribution. Due to Assumption 3.1, the sampler H only requires a random seed of size $O(n^5)$ independently of the parameters a and b, which enable us to connect this theorem to Theorem 2.1 for proving Theorem 3.2. (Similarly to the case of [9], if the size of random seeds depends on a and b, our reduction is in fact not available.)

Now, we briefly explain the differences from previous results. (See Appendix for the details of the proof.) The proof of Theorem 3.3 is done by modifying the arguments used for constructing a hard search problem under some samplable distribution in [7] and a hard decision problem with a derandomization assumption under the uniform distribution in [9]. The essence of these techniques is to find a hard instance against any polynomial-time algorithm A from some formula ϕ_A such as "x is satisfiable but A fails on x" by using A itself.

In our case, we would like to find the hard instance against any search algorithm A by a derandomized version of A with a short random seed. This situation is slightly different from those of [7, 9]. Note that the derandomization is only applicable to decision algorithms, as stated in Assumption 3.1. Therefore, we need to convert the search algorithm A to a decision version B and then we obtain a derandomized decision algorithm B^ε. After that, we construct a derandomized search algorithm from a downward self-reduction with B^ε. Such a downward self-reduction has been already used in [8] and [9].

In the downward self-reduction, the decision algorithm might find some contradiction as follows. Let ψ be some formula obtained by partially assigning to ϕ_A. The downward self-reduction obtain ψ_0 and ψ_1 by assigning 0 and 1 to one variable of ψ and then invokes the satisfiability of ψ_0 and ψ_1 to the decision algorithm. Let us consider the case that the decision algorithm answers "YES" to ψ but "NO" to ψ_0 and ψ_1.

In the arguments of [8] and [9], at least one of ψ, ψ_0, ψ_1 is shown to be a hard instance against the decision algorithm. In this case, we cannot guarantee satisfiability of the hard instance if it is ψ since the decision algorithm answers "YES" on ψ. On the other hand, we require the satisfiability of the hard instance for connecting the hard instance sampler to our reduction (recall that the conditions of Theorem 2.1.) We thus modify the argument of this case.

In our case, A is originally a search algorithm for NP problem. We can then assume that A always answers "NO" on any negative instance without loss of generality since solutions outputted from A are efficiently verifiable, and thus the "YES" answer of A is reliable. The decision algorithm B^ε, constructed from A, inherits a similar property due to the worst-case derandomization assumption. It follows that B^ε correctly answers with high probability on ψ, which implies that the hard instance is at least one of ψ_0 and ψ_1. Since the decision algorithm answers "NO" on them, we can conclude that the hard instance is satisfiable.

Acknowledgements. The authors would like to appreciate Dan Gutfreund for valuable discussions and helpful comments on worst-case to average-case reductions.

References

[1] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 701–710, 2006.

[2] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundation and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[3] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.

[4] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of completes sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993.

[5] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. Technical Report TR95-050, ECCC Report, 1995.

[6] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof system. In Silvio Micali, editor, *Advances in Computing Research, Vol. 5: Randomness and Computation*, pages 73–90. JAI Press, 1989.

[7] Dan Gutfreund. Worst-case vs. average-case complexity in the polynomial-time hierarchy. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2006) and the 10th International Workshop on Randomization and Computation (RANDOM 2006)*, LNCS 4110, pages 386–397, 2006.

[8] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case then it is easy to find their hard instances. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pages 243–257, 2005.

[9] Dan Gutfreund and Amnon Ta-Shma. Worst-case to average-case reductions revisited. In *Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2007) and the 11th International Workshop on Randomization and Computation (RANDOM 2007)*, LNCS 4627, pages 569–583, 2007.

[10] Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006.

[11] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[12] Russell Impagliazzo and Leonid Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 812–821, 1990.

[13] Russell Impagliazzo and Avi Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[14] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):285–286, 1987.

[15] Ryan O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69(1):68–94, 2004.

[16] Rafael Pass. Parallel repetition of zero-knowledge proofs and the possibility of basing cryptography on NP-hardness. In *Proceedings of the 21st IEEE Conference on Computational Complexity*, pages 96–110, 2006.

[17] Luca Trevisan. List decoding using the XOR lemma. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 126–135, 2003.

[18] Luca Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 31–38, 2005.

[19] Andrew C.-C. Yao. Theory and applications of trapdoor functions (extended abstract). In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

Appendix: Construction for Hard Instance Sampler

In this section, we suppose that a decision algorithm outputs either "YES" or "NO" and a search algorithm outputs either a pair of "YES" and a solution or "NO".

We denote by $p_A(x)$ the probability that an algorithm A outputs "YES" on an input x. More precisely, if A is a decision algorithm then $p_A(x) := \Pr_A[A(x) = \text{"YES"}]$, and if A is a search algorithm, $p_A(x) := \Pr_A[A(x) = (\text{"YES"}, \alpha)]$, where α is a solution to the instance x. We sometimes write $A(x) = \text{"YES"}$ instead of $A(x) = (\text{"YES"}, \alpha)$ for a search algorithm A if it succeeds to yield a correct solution.

If A is a search algorithm for SAT in n^a time ($a > 1$), then we can assume that A always outputs "NO" on any unsatisfiable formula x without loss generality since it can check whether the outputted solution indeed satisfies a given formula or not with linear time overhead.

We argue in this section more general derandomization parameterized by δ.

Assumption .4 Let B_1 and B_2 be two randomized decision algorithms. We say that B_1 and B_2 are δ-indistinguishable if $\left|\Pr_{B_1}[B_1(x) = \text{"YES"}] - \Pr_{B_2}[B_2(x) = \text{"YES"}]\right| \leq \delta$ for any $x \in \{0,1\}^*$. For any probabilistic polynomial-time algorithm A and any constant $\varepsilon > 0$, there exists a probabilistic polynomial-time search algorithm A^ε such that A^ε requires a random seed of size at most n^ε on input length n, and A^ε and A are $\delta(n)$-indistinguishable.

Setting $\delta(n) = 1/100$, this assumption is the same one as Assumption 3.1 in Section 3.

We first present a lemma for the hard instance sampler depending on given randomized search algorithms. Using this sampler, we will construct a hard instance sampler stated in Theorem 3.3 against any randomized search algorithm.

Lemma .5 Let $a > 1$ and $b > 0$ be any fixed constants and let $d > 0$ be some constant depending only on a and b. If Assumption .4 holds with a parameter δ and NP $\neq$ RP, for any n^a-time randomized search algorithm A there exists a samplable distribution schema $\mathcal{H}_A = \{\mathcal{H}_{A,n}\}_{n\in\mathbb{N}}$ such that for infinitely many $n \in \mathbb{N}$ and any polynomial $p(\cdot)$

$$\Pr_{x:\mathcal{H}_{A,n}}\left[\Pr_A[A \text{ solves } x] \leq n^{-b} \;\middle|\; x \in \text{SAT}\right] > 1 - 2^{-n^{1/d}} \quad \text{and}$$
$$\Pr_{x:\mathcal{H}_{A,n}}[x \in \text{SAT}] \geq 1/2 - 1/p(n).$$

Then, the sampler $S_{\mathcal{H}_A}$ for $\mathcal{H}^A$ runs in $O(t'(n)\,n^5/\delta')$ time and requires a random seed of size at most $2n^5/\delta'^2$, where $\delta' = \delta + \exp(-n/2)$, $t'(n)$ is a polynomial determined by the derandomization of Assumption .4.

We now give the statement of Theorem 3.3 in Section 3 again.

Theorem .6 Let $a > 1$ and $b > 0$ be any fixed constants and let $d' > 0$ be some constant depending only on a and b. If Assumption .4 holds with a parameter δ and NP $\neq$ RP, there exists a samplable distribution schema $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ such that for any n^a-time randomized search algorithm A for SAT (assignment finding problem), infinitely many $n \in \mathbb{N}$ and any polynomial $p(\cdot)$

$$\Pr_{x:\mathcal{H}_n}\left[\Pr_A[A(x) \text{ yields a solution to } x] \leq n^{-b} \;\middle|\; x \in \text{SAT}\right] > 1 - 2^{-n^{1/d'}} \quad \text{and}$$
$$\Pr_{x:\mathcal{H}_n}[x \in \text{SAT}] \geq 1/2 - 1/p(n).$$

Then, the sampler H for $\mathcal{H}$ runs in $O(t'(n)\,n^5/\delta'^2)$ time and requires a random seed of size at most $2n^5/\delta'^2$, where $\delta' = \delta + \exp(-n/2)$, $t'(n)$ is a polynomial determined by the derandomization of Assumption .4.

We now sketch an overview of the proof of Theorem 3.2. Since we easily obtain the sampler H by modifying a hard instance generator G_A of Lemma .5, we first sketch a rough idea of construction for the sampler G_A.

A basic strategy of our proof follows the technique developed in [8]. For simplification, we only consider a deterministic algorithm here. Roughly speaking, for any decision algorithm B, a hard instance generator G_B performs as follows: (*i*) it first constructs some formula ϕ_B using the *description* of the algorithm B such as "x is satisfiable formula and B fails on x in n^a time." (*ii*) Next, it searches a solution to ϕ_B by using downward self-reduction with the decision algorithm B. (*iii*) If it finds a correct solution x to ϕ_B, It outputs x. If it answers "NO" on ϕ_B, it outputs ϕ_B. If it finds a contradiction in the downward self-reduction, namely, it answers "YES" on $\psi = \phi_B(\alpha_1, ..., \alpha_i, v_{i+1}, ..., v_{n'})$ with a partial assignment $(\alpha_1, ..., \alpha_i) \in \{0,1\}^i$ but it answers "NO" on both of $\psi_0 = \phi_B(\alpha_1, ..., \alpha_i, 0, v_{i+2}, ..., v_{n'})$ and $\psi_1 = \phi_B(\alpha_1, ..., \alpha_i, 1, v_{i+2}, ..., v_{n'})$, it then outputs one of these three formulae ψ, ψ_0, ψ_1 uniformly at random.

Intuitively, either x, ϕ_B or one of ψ, ψ_0, ψ_1 is hard against B, so is it against A. The reason is: (*i*) If x is found then B should fail on x as stated in the formula. (*ii*) The worst-case assumption NP $\neq$ P implies that ϕ_B is a satisfiable formula since there is a hard instance x for B by the assumption. So, if B answers "NO" on ϕ_B, B indeed fails on ϕ_B. (*iii*) Apparently, B fails on at least one of three formulae ψ, ψ_0, ψ_1. Therefore, the algorithm B itself provides a hard instance against B with the input ϕ_B.

The results of [7, 9] also make use of the basic idea of [8]. The result of [7] provides a version of a distribution hard against randomized search algorithms without the derandomization, and the result of [9] provides another version against decision algorithms with a similar derandomization to ours.

On the other hand, we treat a distribution hard against randomized *search* algorithms with the *mild derandomization assumption* unlike the cases of [8, 7, 9]. So, we require a different technical argument from theirs.

Recall that the mild derandomization of Assumption 3.1 is applicable only to decision algorithms. To construct the derandomized search algorithm A^ε from the original search algorithm A, we regard A as a decision algorithm B once and then derandomize it to a decision algorithm B^ε that approximates B well. After that, we make use of the derandomized decision algorithm B^ε as a decision oracle to perform the downward self-reduction for SAT instances.

So, we obtain three cases such as [8] (and [9]): the generator finds x to ϕ_A, answers "NO" on ϕ_A, or finds a contradiction in the downward self-reduction. Note that we require satisfiability of hard instances outputted from the generator (with a nonnegligible probability) to apply Theorem 2.1 to this worst-case to average-case reduction. (See the condition of Theorem 2.1.)

The first two cases satisfy this requirement even in the case of the basic argument of [8]. However, the third case is problematic. It is because that the algorithm wrongly outputs "YES" on the self-reduced instance ψ and if this is the hard instance then the hard instance is not satisfiable.

Fortunately, we can avoid this problem in our setting by the fact that A is originally a search algorithm. This fact implies that if the decision algorithm B^ε answers "YES" then the answer is correct with high probability. Note that a solution outputted from NP-search algorithm A is efficiently verifiable, and thus the "YES" answer of A is reliable. This property is inherited by B and hence B^ε due to the worst-case derandomization assumption. (This is the reason that we require the "worst-case" derandomization assumption. Only assuming an average-case derandomization, even if B^ε inherits the property of B on some instance x, we cannot guarantee that B^ε does on a self-reduced instance of x.)

Therefore, when B^ε outputs "YES" on ψ, the answer is correct with high probability. Then at least one of ψ_0 and ψ_1 is the hard instance, say ψ_i, on which B^ε answers "NO". Since ψ_i is hard and B^ε answers "NO" on it, ψ_i should be satisfiable with high probability. It follows that the generator can output a hard satisfiable instance at least approximately $1/2$ in this case.

Finally, we roughly explain how to convert this hard instance generator for a specific algorithm to one for any algorithm. For this conversion, we just modify the statement of the base formula "x is satisfiable and A fails on x in n^a time" to "x is satisfiable and all of $A_1, ..., A_{\log\log n}$ fail on x in n^a time", where $A_1, ..., A_{\log\log n}$ are the first $\log\log n$ randomized algorithms. The proof of this case goes through similarly to the previous case. Note that any algorithm is included in the set of $\{A_1, ..., A_{\log\log n}\}$ for all sufficiently large n. We can therefore conclude that the modified sampler generates hard

instances against any n^a-time randomized algorithm for such large n.

A Proof of Lemma .5

In what follows, we first give the base formula ϕ (Subsection A.1) and our sampling algorithm H_A generating the distribution schema H_A (Subsection A.3), and then we prove that H_A indeed generates hard instances against A (Subsection A.4).

A.1 Construction for Base Formula

The following is our base formula:

$$\phi_{n,r} \equiv \exists x \in \{0,1\}^n \left[\mathrm{SSAT}^1(x; r) = \text{“NO”} \wedge x \in \mathrm{SAT} \right],$$

where SSAT^1 is a randomized search algorithm for SAT, which is constructed from the original algorithm A and Assumption .4.

We assume that $\phi_{n,r}$ directly represents a formula of a SAT instance by the Cook-Levin reduction for each fixed n and r. The length of $\phi_{n,r}$ is then bounded by some polynomial if SSAT^1 runs in polynomial time.

To construct the search algorithm SSAT^1, we make use of a derandomized decision algorithm $\overline{\mathrm{BSAT}}^1$. Here we consider a general algorithm $\overline{\mathrm{BSAT}}^\varepsilon$ with a parameter $\varepsilon > 0$. (Setting $\varepsilon = 1$, we obtain the algorithm $\overline{\mathrm{BSAT}}^1$.) This algorithm $\overline{\mathrm{BSAT}}^\varepsilon$ has the following properties.

Lemma A.1 Suppose that Assumption .4 holds. For any randomized search algorithm A running in n^a time, there exists a randomized decision algorithm $\overline{\mathrm{BSAT}}^\varepsilon$ such that

1. Given a formula x of length n, the running time and the size of random seeds of $\overline{\mathrm{BSAT}}^\varepsilon$ are $O(t(n)\, n^2/\delta'^2)$ and $n^3/2\delta'^2$ respectively, where $t(n)$ denotes a polynomial determined by Assumption .4 to reduce the size of random seeds from n^{a+b+1} to n and $\delta' = \delta + \exp(-n/2)$.

2. We have for any x

$$\begin{aligned} x \in \mathrm{SAT} \ \wedge \ p_A(x) > n^{-b}/2 \quad &\Rightarrow \quad p_{\overline{\mathrm{BSAT}}^\varepsilon}(x) > 1 - \exp(-n^{2\varepsilon}) \quad \text{and} \\ x \notin \mathrm{SAT} \quad &\Rightarrow \quad p_{\overline{\mathrm{BSAT}}^\varepsilon}(x) \le \exp(-n^{2\varepsilon}). \end{aligned}$$

To show these properties, we require several intermediate algorithms and their analysis. For ease of understanding, we put the proof to Subsection A.2.

By using $\overline{\mathrm{BSAT}}^\varepsilon$ with $\varepsilon = 1$, we obtain the search algorithm SSAT^1 as follows:

Algorithm: SSAT^1(input $x = x(v_1, ..., v_{n'})$)

1. Search a satisfying assignment by a downward self-reduction using $\overline{\mathrm{BSAT}}^1(\cdot)$ as a decision oracle for SAT. Precisely, perform the following steps:

(i) Run $\overline{\mathrm{BSAT}}^1(x)$. If the result is "NO", output "NO" and halt.

(ii) Set $i = 0$ and $x^{(0)} = x$ initially. Let $x^{(i)} = x(\alpha_1, ..., \alpha_i, v_{i+1}, ..., v_{n'})$ be the current formula obtained by partially assigning an i-bit string $(\alpha_1, ..., \alpha_i)$ to the variables $v_1, ..., v_i$ in x.

(iii) Assign 0 and 1 to the variable v_{i+1} in $x^{(i)}$. Let $x_0^{(i+1)}$ and $x_1^{(i+1)}$ be the resulting formulae with $v_{i+1} = 0$ and 1 respectively.

(iv) Run $\overline{\mathrm{BSAT}}^1(x_0^{(i+1)})$ and $\overline{\mathrm{BSAT}}^1(x_1^{(i+1)})$. If both of the results are "NO", output "ERROR" and halt. If at least one of them is "YES", update the current formula to it, and go back to Step (ii).

2. If a satisfying assignment for x is found in Step 1, output "YES" and the assignment.

Straightforwardly, we obtain the following lemma from Lemma A.1 and the above construction.

Lemma A.2 Suppose that Assumption .4 holds. For any randomized search algorithm A running in n^a time, there exists a randomized decision algorithm $\overline{\mathrm{BSAT}}^{\varepsilon}$ such that

1. Given a formula x of length n, the running time and the size of random seeds of SSAT^1 are at most $O(t(n)\, n^3/\delta'^2)$ and n^4/δ'^2 respectively, where $t(n)$ denotes a polynomial determined by Assumption .4 to reduce the size of random seeds from n^{a+b+1} to n and $\delta' = \delta + \exp(-n/2)$.

2. We have for any x

$$\begin{aligned} x \in \mathrm{SAT} \ \wedge \ p_A(x) > n^{-b}/2 \quad &\Rightarrow \quad \Pr\left[\, \mathrm{SSAT}^1(x) \neq \text{"NO"} \,\right] > 1 - \exp(-n^2) \quad \text{and} \\ x \notin \mathrm{SAT} \quad &\Rightarrow \quad p_{\mathrm{SSAT}^1}(x) = 0. \end{aligned}$$

A.2 Modified Decision Algorithm

Let A be a search algorithm that tries to solve SAT. Recall that A outputs a pair of "YES" and a satisfying assignment if it succeeds, otherwise it just outputs "NO". As already mentioned, we can assume A always outputs "NO" on an input $x \notin \mathrm{SAT}$ without loss of generality since A is a search algorithm for SAT.

For our base formula, we require a decision algorithm $\overline{\mathrm{BSAT}}^1$ that works correctly with very high probability if A does with probability at least n^{-b} with a short random seed.

Let BSAT be a decisional version of a given search algorithm A of running time n^a. BSAT outputs "YES" (success) or "NO" (failure) depending on the output of A. Thus the performance (time, size of random seed, and success probability) of BSAT is the same as that of A, and BSAT outputs "NO" on an input $x \notin \mathrm{SAT}$ with probability 1.

We construct the base formula we desire by modifying this decision algorithm BSAT. We first introduce an amplified version of BSAT.

Algorithm: BSAT_b (input x)

1. Choose $v_1, ..., v_{n^{b+1}} \in \{0,1\}^{n^a}$ independently and uniformly at random.

2. Run $\mathrm{BSAT}(x; v_i)$ for $i = 1, ..., n^{b+1}$.

3. Output "YES" if there is at least one "YES" in the results of Step 2. Otherwise, output 0.

Obviously, the running time is $O(n^{a+b+1})$ and the size of random seeds is at most n^{a+b+1}. Now we estimate the success probability of BSAT_b.

Claim 6 For any $x \in \mathrm{SAT}$, if $p_{\mathrm{BSAT}}(x) > n^{-b}/2$ then $p_{\mathrm{BSAT}_b}(x) > 1 - \exp(-n/2)$. For any $x \notin \mathrm{SAT}$, $p_{\mathrm{BSAT}_b}(x) = 0$.

The first statement on the probability estimation can be derived by a standard argument using the Hoeffding bounds. The second one is done by the fact that BSAT does not fail on a negative instance.

We next apply Assumption .4 to BSAT_b for saving randomness. We denote the resulting algorithm by $\mathrm{BSAT}^\varepsilon$. The algorithm $\mathrm{BSAT}^\varepsilon$ requires a random seed of size at most n^ε to simulate BSAT_b, and runs in $t_\varepsilon(n)$ time, where $t_\varepsilon(n)$ is a polynomial determined by the derandomization of Assumption .4. Then $\mathrm{BSAT}^\varepsilon$ and BSAT_b are δ-indistinguishable.

Claim 7 Let $\delta' := \delta + \exp(-n/2)$. Then $\mathrm{BSAT}^\varepsilon$ satisfies the following:

1. For $x \in \mathrm{SAT}$, if $p_{\mathrm{BSAT}_b}(x) > 1 - \exp(-n/2)$ then $p_{\mathrm{BSAT}^\varepsilon}(x) > 1 - \delta'$.
2. For $x \notin \mathrm{SAT}$, $p_{\mathrm{BSAT}^\varepsilon}(x) \leq \delta$.

Now we amplify the gap between the probabilities by taking majority vote. This is the decision algorithm $\overline{\mathrm{BSAT}}^\varepsilon$ we require.

Algorithm: $\overline{\mathrm{BSAT}}^\varepsilon$(input x)

1. Choose $v_1, ..., v_{n^{2\varepsilon}/2\delta'^2} \in \{0,1\}^n$ independently and uniformly at random for $\delta' = \delta + \exp(-n/2)$.
2. Run $\mathrm{BSAT}^\varepsilon(x; v_i)$ for $i = 1, ..., n^{2\varepsilon}/2\delta'^2$.
3. Output the majority of the results in Step 2.

Claim 8 Then the algorithm $\overline{\mathrm{BSAT}}^\varepsilon$ satisfies the following:

1. For $x \in \mathrm{SAT}$, if $\mathrm{BSAT}^\varepsilon(x) > 1 - \delta'$ then $\overline{\mathrm{BSAT}}^\varepsilon(x) > 1 - \exp(-n^{2\varepsilon})$.
2. For $x \notin \mathrm{SAT}$, $p_{\overline{\mathrm{BSAT}}^\varepsilon}(x) \leq \exp(-n^{2\varepsilon})$.

The running time and the size of random seeds of $\overline{\mathrm{BSAT}}^\varepsilon$ are $O(t_\varepsilon(n)\, n^{2\varepsilon}/\delta'^2)$ and $n^{3\varepsilon}/2\delta'^2$ respectively.

Putting the above claims together, we can obtain the proof of Lemma A.1.

A.3 Construction for Hard Instance Sampler

We first give G_A that generates hard instances by using the base formula $\phi_{n,r}$. The algorithm G_A is constructed from another search algorithm $\mathrm{SSAT}^\varepsilon$ for SAT, which is obtained from $\overline{\mathrm{BSAT}}^\varepsilon$. The algorithm $\mathrm{SSAT}^\varepsilon$ works as a component of the sampler to find a solution to the base formula $\phi_{n,r}$ for generating a hard instance. In this algorithm, we fix the parameter ε to $1/d$ for the length n^d of $\phi_{n,r}$.

Algorithm: $\mathrm{SSAT}^{\varepsilon}(\text{input } \phi_{n,r} = \phi_{n,r}(v_1, ..., v_{n'}))$

1. Choose $u \in \{0,1\}^{(n^d)^{3\varepsilon/2\delta'^2}} = \{0,1\}^{n^{3/2\delta'^2}}$ uniformly at random.

2. Search a satisfying assignment by a downward self-reduction using $\overline{\mathrm{BSAT}}^{\varepsilon}(\cdot; u)$ as a decision oracle for SAT. Precisely, perform the following steps:

(i) Run $\overline{\mathrm{BSAT}}^{\varepsilon}(\phi_{n,r}; u)$. If the result is "NO", output $\phi_{n,r}$ and halt.

(ii) Set $i = 0$ and $\psi^{(0)} = \phi_{n,r}$ initially. Let $\psi^{(i)} = \phi_{n,r}(\alpha_1, ..., \alpha_i, v_{i+1}, ..., v_{n'})$ be the current formula obtained by partially assigning an i-bit string $(\alpha_1, ..., \alpha_i)$ to the variables $v_1, ..., v_i$ in $\phi_{n,r}$.

(iii) Assign 0 and 1 to the variable v_{i+1} in $\psi^{(i)}$ and then pad them to be of length n^d. Let $\psi_0^{(i+1)}$ and $\psi_1^{(i+1)}$ be the resulting formulae with $v_{i+1} = 0$ and 1 respectively.

(iv) Run $\overline{\mathrm{BSAT}}^{\varepsilon}(\psi_0^{(i+1)}; u)$ and $\overline{\mathrm{BSAT}}^{\varepsilon}(\psi_1^{(i+1)}; u)$. If both of the results are "NO", output "ERROR" and $(\psi_0^{(i+1)}, \psi_1^{(i+1)})$ and halt. If at least one of them is "YES", update the current formula to it, and go back to Step (ii).

3. If a satisfying assignment for $\phi_{n,r}$ is found in Step 2, output "YES" and the assignment.

Note that there are three possibilities of outputs from $\mathrm{SSAT}^{\varepsilon}$: ("YES", x), ("ERROR", $(\psi_0^{(i+1)}, \psi_1^{(i+1)})$) and ("NO", $\phi_{n,r}$). Then, $|x| = n$, $|\psi_0^{(i+1)}| = |\psi_1^{(i+1)}| = |\phi_{n,r}| = n^d$. Moreover, this $\mathrm{SSAT}^{\varepsilon}$ has the following properties, which are easily proven by construction for $\mathrm{SSAT}^{\varepsilon}$ and Lemma A.1.

Lemma A.3 Suppose that Assumption .4 holds. The algorithm $\mathrm{SSAT}^{\varepsilon}$ satisfies the following:

1. Given a formula $\phi_{n,r}$ of length n^d, the running time and the size of random seeds of $\mathrm{SSAT}^{\varepsilon}$ are $O(t'(n)\, n^{3d}/\delta'^2)$ and n^4/δ'^2 respectively, where $t'(n)$ denotes a polynomial determined by Assumption .4 to reduce the size of random seeds from $n^{d(a+b+1)}$ to $n^{1/d}$ and $\delta' = \delta + \exp(-n/2)$.
2. We have for any y of length n^d

$$y \in \mathrm{SAT} \ \wedge \ p_A(y) > |y|^{-b}/2 \ \Rightarrow \ \Pr_{\mathrm{SSAT}^{\varepsilon}}[\, \mathrm{SSAT}^{\varepsilon}(y) \neq \text{"NO"} \,] > 1 - |y| \exp(-|y|^2)$$

Combining the above algorithm $\mathrm{SSAT}^{\varepsilon}$ with the formula $\phi_{n,r}$, we now show a hard instance generator G_A as follows.

Algorithm: G_A (input 1^n)

1. Run $\mathrm{SSAT}^{\varepsilon}(\phi_{n,r}; u)$ for $r \in \{0,1\}^{n^4/\delta'^2}$ and $u \in \{0,1\}^{n^3/2\delta'^2}$ chosen independently and uniformly at random.

2. If it obtains ("YES", x), ("ERROR", ψ_0, ψ_1) and ("NO", ϕ), output x, (ψ_0, ψ_1), ϕ, respectively.

Note that the output size is different from each other among "YES", "ERROR" and "NO". Thus, we modify this algorithm G_A to use it as a sampler H_A for the distribution schema H_A by a similar argument to [8, 7, 9]. The following is the modified algorithm, i.e., the sampler H_A for $\mathcal{H}$.

Algorithm: H_A (input 1^n)

1. Run $G_A(1^n)$ n times independently. If it outputs an instance of length n, then output it and halt.

2. Check whether n satisfies $m^d = n$ for some m, where $|\phi_{n,r}| = n^d$. If there is such an m, run $G_A(1^m)$ n times independently. If it outputs an instance of length n, then output it. If it outputs a pair of instances of length n, output one of them uniformly at random.

3. Output 0^n.

A.4 Correctness

We first prove that the generator G_A provides hard instances against the original search algorithm A.

Lemma A.4 The algorithm G_A satisfies the following:

1. If $G_A(1^n)$ outputs "YES" and x of length n satisfying $\phi_{n,r}$, we have
$$x \in \mathrm{SAT} \quad \text{and} \quad \Pr_{G_A}\left[\, p_A(x) \le n^{-b}/2 \right] > 1 - 2^n \exp(-n^2).$$
2. If $G_A(1^n)$ outputs "NO" and y of length n^d, we have
$$\Pr_{G_A}[\, y \in \mathrm{SAT}\,] > 1 - 1/p(n) \quad \text{and} \quad \Pr_{G_A}\left[\, p_A(y) \le |y|^{-b}/2 \;\middle|\; y \in \mathrm{SAT} \right] > 1 - 2^{|y|}\exp(-|y|^2)$$
for any polynomial $p(\cdot)$.
3. If $G_A(1^n)$ outputs "ERROR" and (ψ_0, ψ_1) of length n^d, there exits a formula $z \in \{\psi_0, \psi_1\}$ such that
$$\Pr_{G_A}[\, z \in \mathrm{SAT}\,] > 1 - p(n) - |z|\exp(-|z|^2) \quad \text{and} \quad \Pr_{G_A}\left[\, p_A(z) \le |z|^{-b}/2 \;\middle|\; z \in \mathrm{SAT} \right] > 1 - 2^{|z|}\exp(-|z|^2).$$

Proof. **Case 1** ($\mathrm{SSAT}^\varepsilon$ outputs "YES" on $\phi_{n,r}$): In this case $\mathrm{SSAT}^\varepsilon$ yields a correct solution x to $\phi_{n,r}$ at the same time. Then, x is a satisfiable formula and $\mathrm{SSAT}^1(x; r)$ = "NO" from the statement of $\phi_{n,r}$. We now give the following claim on the random string r.

Claim 9 We say that $r \in \{0,1\}^{n^4/\delta'^2}$ is good for SSAT^1 if we have for any x of length n $p_A(x) > n^{-b}/2 \Rightarrow \mathrm{SSAT}^1(x; r) \neq$ "NO". We then have
$$\Pr_{r:\mathcal{U}_{n^4/\delta'^2}}\left[\, r \text{ is good for } \mathrm{SSAT}^1 \,\right] > 1 - 2^n \exp(-n^2).$$

Proof. By Lemma A.2, if $p_A(x) > n^{-b}/2$, a fraction of r's satisfying $\mathrm{SSAT}^1(x; r)$ = "NO" is at most $\exp(-n^2)$. It follows that a fraction of bad r's for a fixed x is at most $\exp(-n^2)$. By taking the union bound over all valid instances, we obtained the desired bound for good r's. □ Taking the contraposition of the definition of a good r, we obtain that $p_A(x) \le n^{-b}/2$ for a satisfiable x.

Case 2 ($\mathrm{SSAT}^\varepsilon$ outputs "NO" on $\phi_{n,r}$): We first show that $\phi_{n,r}$ is satisfiable for almost all r's under the worst-case assumption NP ≠ RP.

Claim 10 Assume that NP ≠ RP. Then, for any polynomial $p(\cdot)$, we have
$$\Pr_r[\, \phi_{n,r} \in \mathrm{SAT}\,] > 1 - 1/p(n).$$

Proof. Recall that any search algorithm for SAT can be assumed to answer correctly on any negative instance. If NP ≠ RP, for any polynomial $p(\cdot)$ there exists a satisfiable formula x such that $p_{\mathrm{SSAT}^1}(x) \le 1/p(n)$ since SSAT^1 is a polynomial-time NP-search algorithm. This implies that for $1 - 1/p(n)$ fraction of r's $\mathrm{SSAT}^1(x; r)$ = "NO". It follows that the formula "$\mathrm{SSAT}^1(x; r)$ = "NO" $\wedge$ $x \in \mathrm{SAT}$" is satisfied by these x and r, i.e., $\phi_{n,r} \in \mathrm{SAT}$. □

By Lemma A.3 and the same argument as the proof of Claim 9, we obtain the following claim.

Claim 11 We say that $u \in \{0,1\}^{(n^d)^3/2\delta'^2}$ is good for $\mathrm{SSAT}^\varepsilon$ if we have $p_A(x) > |y|^{-b}/2 \Rightarrow \mathrm{SSAT}^\varepsilon(x; u) \neq$ "NO" for any y of length n^d. We then have
$$\Pr_{r:\mathcal{U}_{|y|^3/2\delta'^2}}[\, u \text{ is good for } \mathrm{SSAT}^\varepsilon\,] > 1 - 2^{|y|}\exp(-|y|^2).$$

By taking the contraposition, it follows that $p_A(\phi_{n,r}) \leq |\phi_{n,r}|^{-b}/2$ since SSAT$^\varepsilon$ outputs "NO" in this case.

Case 3 (SSAT$^\varepsilon$ outputs "ERROR" on $\phi_{n,r}$): Then, SSAT$^\varepsilon$ outputs two formulae $\psi_0^{(i+1)}$ and $\psi_1^{(i+1)}$ for some i at the same time. Let $\psi^{(i)}$ be the formula before assigning to the variable v_{i+1} in Step (*iii*) of SSAT$^\varepsilon$. We first suppose that $\psi^{(i)}$ is satisfiable. We then have a formula $\phi^{(i+1)} \in \{\psi_0^{(i+1)}, \psi_1^{(i+1)}\}$ such that $\phi^{(i+1)}$ is satisfiable but $\overline{\text{BSAT}}^\varepsilon$ outputs "NO" on $\phi^{(i+1)}$. By the definition of a good u, if u is good for $\overline{\text{BSAT}}^\varepsilon$, it follows that $p_A(\phi^{(i+1)}) \leq |\phi^{(i+1)}|^{-b}/2$.

Next, we argue that $\psi^{(i)}$ is satisfiable with high probability until $\overline{\text{BSAT}}^\varepsilon$ outputs "NO" on both of $\psi_0^{(i+1)}$ and $\psi_1^{(i+1)}$. Suppose that the starting formula $\phi_{n,r}$ is satisfiable. By Lemma A.1, we have $\overline{\text{BSAT}}^\varepsilon(y)$ = "YES" with probability at most $\exp(-|y|^{2\varepsilon})$ for any $y \notin$ SAT. Since the number of the variables in $\phi_{n,r}$ is at most n^d, the unsatisfiable formula is chosen in the downward self-reduction, , i.e., $\psi^{(i)}$ is unsatisfiable for some i with probability at most $n^d \exp(-n^{2d\varepsilon})$ by the union bound. □

We now turn to the correctness of the sampler H_A.

Proof of Lemma .5. We prove that the instances outputted in either Steps 1 or 2 of H_A are hard against A for infinitely many n.

We consider three cases in the analysis of H_A. The first case is that H_A outputs an instance in Step 1. This instance is outputted only if SSAT$^\varepsilon$ in G_A outputs "YES", that is, it succeeds to yield a solution x to $\phi_{n,r}$. The second one is H_A outputs an instance in Step 1 when SSAT$^\varepsilon$ in G_A outputs "NO". The third one is when SSAT$^\varepsilon$ in G_A outputs "ERROR". Below we call these outputs from H_A yes-, no-, and error-instances, respectively. Note that outputs of G_A are always of length n.

We define sets N and N' as

$$N := \left\{ n \in \mathbb{N} : \Pr_{G_A}[\, G_A \text{ outputs } x \text{ of length } n \text{ in Step 1} \,] > 1/2 \right\} \quad \text{and} \quad N' := \{m^d : m \in \mathbb{N}\} \setminus N. \tag{5}$$

Note that at least one of N and N' is an infinite set. We will show that if $n \in N$ then an yes-instance is hard and if $n \in N'$ then either no- or error-instance is hard.

We begin with the following property on length parameter n. By this property, one can see that H_A outputs instances in Steps 1 and 2 with high probability for infinitely many n.

Claim 12 We have

$$\Pr_{H_A}[\, H_A \text{ outputs an instance in Step 1} \,] > 1 - 2^{-n}$$

for any $n \in N$, and we also have

$$\Pr_{H_A}[\, H_A \text{ outputs an instance in Step 2} \,] > 1 - 2^{-n}$$

for any $n \in N'$.

Proof. By the definition of N in Equation (5), the generator G_A outputs a solution x to $\phi_{n,r}$ with probability at least $1/2$ for any $n \in N$ and outputs $\phi_{n,r}$ with probability at least $1/2$ for any $n \in N'$. By repeating G_A n times in Steps 1 and 2 of H_A, H_A obtains an instance in Step 1 for $n \in N$, and an instance in Step 2 for $n \in N'$ with probability at least $1 - 2^{-n}$. □

We suppose that n is in the set N for Case 1 and $n = m^d$ is in the set $N' = \{m^d : m \in \mathbb{N}\} \setminus N$ in Cases 2 and 3. By Claim 12, H_A outputs an yes-instance for $n \in N$ and outputs no- or error-instance for $n \in N'$ with probability at least $1 - 2^{-n}$.

Case 1: We now discuss Case 1 where the sampler outputs an yes-instance for every $n \in N$ defined in Equation (5). In this case, we obtain a satisfying assignment x of length n for $\phi_{n,r}$ from the property of SSAT$^\varepsilon$, which concludes

$$x \in \text{SAT} \quad \text{and} \quad \Pr_{G_A}\left[\, p_A(x) \leq n^{-b}/2 \right] > 1 - 2^n \exp(-n^2)$$

by Lemma A.4 1.

Case 2: Next, we consider the case that H_A outputs a no-instance. It is obvious that for the instance y

$$\Pr_{G_A}[\, y \in \mathrm{SAT}\,] > 1 - 1/p(n) \quad \text{and} \quad \Pr_{G_A}\Big[\, p_A(y) \leq |y|^{-b}/2 \;\Big|\; y \in \mathrm{SAT}\,\Big] > 1 - 2^{|y|}\exp(-|y|^2)$$

for any polynomial $p(\cdot)$ by Lemma A.4 2.

Case 3: In this case, we obtain two instances ψ_0 and ψ_1 in $\mathrm{SSAT}^{\varepsilon}$. By Lemma A.4 3, one of two instances is satisfiable and hard against A. Since H_A chooses one of two uniformly at random, if H_A chooses $z \in \{\psi_0, \psi_1\}$ we have

$$\Pr_{H_A}[\, z \in \mathrm{SAT}\,] > 1/2 - p(n) - |z|\exp(-|z|^2)/2 \quad \text{and} \quad \Pr_{H_A}\Big[\, p_A(z) \leq |z|^{-b}/2 \;\Big|\; z \in \mathrm{SAT}\,\Big] > 1 - 2^{|z|}\exp(-|z|^2).$$

for any polynomial $p(\cdot)$.

Putting Cases 1, 2 and 3 together, we can demonstrate that H_A is indeed a hard instance sampler for infinitely many n, as stated in Lemma .5. □