Electronic Colloquium on Computational Complexity, Report No. 23 (2009)

Strong Hardness Preserving Reduction from a P-Samplable Distribution to the Uniform Distribution for NP-Search Problems

Akinori Kawachi and Osamu Watanabe
Dept. of Math. & Comp. Sci., Tokyo Inst. of Tech, Tokyo
{kawachi,watanabe}(at)is.titech.ac.jp

Abstract. Impagliazzo and Levin demonstrated [IL90] that the average-case hardness of any NP-search problem under any P-samplable distribution implies that of another NP-search problem under the uniform distribution. For this they developed a way to define a reduction from an NP-search problem F with "mild hardness" under any P-samplable distribution $\mathcal{H}$; more specifically, F is a problem with positive hard instances with probability $1/\mathrm{poly}(n)$ under $\mathcal{H}$. In this paper we show a similar reduction for an NP-search problem F with "strong hardness", that is, F with positive hard instances with probability $1 - 1/\mathrm{poly}(n)$ under $\mathcal{H}$ in its positive domain (i.e., the set of positive instances). Our reduction defines from this pair of F and $\mathcal{H}$, some NP-search problem G with a similar hardness under the uniform distribution $\mathcal{U}$; more precisely, (i) G has positive hard instances with probability $1 - 1/\mathrm{poly}(n)$ under $\mathcal{U}$ in its positive domain, and (ii) the positive domain itself occupies $1/\mathrm{poly}(n)$ of $\{0,1\}^n$.

1 Introduction

The theory of the average-case complexity has been studied extensively since 1970's, and during late 80's to early 90's, several fundamental results have been shown (see an excellent survey [BT06b] for the background). Among such results, an average-case NP-completeness theorem shown by Impagliazzo and Levin [IL90] is one of the seminal results in the average-case computational complexity theory. They demonstrated some NP problem Y whose hardness under the uniform distribution $\mathcal{U}$ is complete in the sense that it is essentially harder than any NP problem X under any polynomial-time samplable distribution $\mathcal{H}$. In high level, they showed a reduction from $(X, \mathcal{H})$ to $(Y, \mathcal{U})$ that keeps certain hardness. Here we propose a reduction showing a closer complexity relation for investigating stronger hardness properties.

From some technical reason (see discussion in the last section), we consider throughout this paper only *NP-search problems*, i.e., problems for finding a polynomial-time verifiable witness for a given *positive instance* (i.e., "yes" instance) of the corresponding NP decision problem. In this context the Impagliazzo-Levin reduction is stated as follows. Let F be any NP-search problem, and let L be its *positive domain*, i.e., the set of its positive instances. Let $\mathcal{H}$ be a polynomial-time samplable distribution for inputs under which we would like to investigate the complexity of the search problem F. Then for this pair of F and $\mathcal{H}$, their reduction defines some NP-search problem G with a positive domain M that satisfies the following property: Suppose that some polynomial-time (randomized) algorithm B succeeds to solve G with high probability under the uniform distribution over its positive domain M. More specifically, suppose that

ISSN 1433-8092

$\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}[\,B(\overline{x}) \in G(\overline{x}) \mid \overline{x} \in M\,] > 1 - \delta(n)$ holds for some $\delta(n) > 0$ (where n and $\overline{n}$ are polynomially related). Then we have some algorithm A satisfying $\Pr_{x:\mathcal{H}_n}[\,A(x) \in F(x) \mid x \in L\,] > 1 - \mathrm{poly}(n) \cdot \delta(n)$. With reductions of this type, we can show that the average-case NP-hardness under any polynomial-time samplable distribution is essentially equivalent to the one under the uniform distribution (see, e.g., [BT06b] for an exact statement and its proof).

Roughly speaking, the above reduction shows that F's hard part is not reduced to more than $1/\mathrm{poly}(n)$ in G; that is, if F's hard part occupies (measured under $\mathcal{H}$) $\mathrm{poly}(n) \cdot \delta(n)$ of its positive domain, then the proportion of G's hard part in its positive domain is at least $\delta(n)$. In applications of the average-case complexity theory, we sometimes require higher hardness, e.g., hardness on all but a small fraction of instances. Such examples can be found in cryptographic applications. The above reduction from $(F, \mathcal{H})$ to $(G, \mathcal{U})$ is not appropriate for investigating such strong hardness properties. For example, even if we know that the proportion of F's hard part is close to 1 under $\mathcal{H}$, it only guarantees some $1/\mathrm{poly}(n)$ lower bound for the proportion of G's hard part. We need a reduction that guarantees that easy part proportion does not get enlarged from the original problem to the reduced problem. Unfortunately, however, the technique used in the Impagliazzo-Levin reduction does not seem appropriate for this purpose (see more technical explanation in the later section). In this paper, we propose some techniques added on the one used in the Impagliazzo-Levin reduction to design a new reduction for providing such closer hardness relationship.

Let us state our main result precisely. For this and for our later discussion, we first introduce some notions and notations. For any NP-search problem F and any instance x, we use $F(x)$ to denote the set of solutions of F on x. We assume that each solution for x is expressed in $\{0,1\}^*$ of length polynomially bounded by $|x|$ and that some deterministic algorithm checks whether a given w is a correct solution (i.e., $w \in F(x)$ or not) within polynomial-time in $|x|$. An instance x is called *positive* if $F(x) \neq \emptyset$.

In order for discussing the hardness of a given problem, we consider input instance distributions. A distribution is specified as an ensemble $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 0}$, where $\mathcal{H}_n$ is a distribution restricted on $\{0,1\}^n$. We use $\mathcal{U} = \{\mathcal{U}_n\}_{n \geq 0}$ to denote the uniform distribution. By "$x : \mathcal{H}_n$" and "$\mathcal{H}_n(x)$", we mean respectively a random instance x occurred according to $\mathcal{H}_n$ and the probability that x occurs according to $\mathcal{H}_n$. A distribution $\mathcal{H}$ is called *polynomial-time samplable* if there exists some polynomial-time randomized algorithm or a *sampler* H such that $H(1^n)$ produces each $x \in \{0,1\}^n$ with probability $\mathcal{H}_n(x)$. We assume that $H(1^n)$ always consumes a random seed of the same length, say, n^s. We use $H(1^n; r)$ to denote the execution of H on an input 1^n and a random seed $r \in \{0,1\}^{n^s}$.

Now we are ready to state our main theorem. We will prove the following theorem.

Theorem 1. We have some constants c_0, d_0, and e_0 with which the following statement holds. Let F be any NP-search problem with a positive domain L, and let $\mathcal{H}$ and H be any polynomial-time samplable distribution for F and its polynomial-time sampler respectively. Suppose that F, $\mathcal{H}$, and H satisfy the following with some constants $d \geq 1$ and $s \geq 1$.

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 exist some NP-search problem G and positive domain M that satisfy the following.

4. $\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}[\ \overline{x} \in M\ ] \ \geq\ (\overline{n})^{-(d_0+d)}$,
5. if some $(\overline{n})^t$-time randomized search algorithm B for G achieves

$$\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}[\ \Pr_{B}[B(\overline{x}) \text{ yields some solution} \in G(\overline{x})\,] \ \geq\ 1/2 \ \mid\ \overline{x} \in M\ ] \ \geq\ (\overline{n})^{-e}, \qquad (1)$$

then by using B one can define some $n^{c_0 \cdot s \cdot t}$-time randomized search algorithm A that solves F in the following sense:

$$\Pr_{x:\mathcal{H}_n}[\ \Pr_{A}[A(x) \text{ yields some solution} \in F(x)\,] \ \geq\ n^{-e_0 \cdot s \cdot e} \ \mid\ x \in L\ ] \ \geq\ n^{-e_0 \cdot s \cdot e}. \qquad (2)$$

Remark. Throughout this paper, we will use n and $\overline{n}$ to denote respectively the length of instances for F and G. As we will see in the proof, $\overline{n}$ is polynomially bounded by n. Constants c_0, d_0, and e_0 are determined by choice of hash function families and a tupling function (used in the proof) and do not depend on size parameters, distributions, nor NP-search problems. Such constants will be called *universal constants* in the following.

There may be some other ways to show a similar relationship. For example, one might consider using direct product techniques, i.e., Yao's XOR lemma [Yao82] and its powerful variants. These techniques are quite useful for hardness amplification. Thus, we may be able to use the Impagliazzo-Levin reduction and then amplify its hardness by one of those direct product techniques. This approach, however, seems to have some disadvantage in our setting. They basically make an instance of a harder problem by taking direct product of multiple instances of the original problem, which is some NP-search problem in our case. But then fraction of positive instances would decrease exponentially, and this may not be appropriate for investigating strong hardness properties. One may be able to apply some hash function to increase the fraction of positive instances in its domain; but it is not so obvious to find an appropriate way. Here we rather take a more direct approach for designing a reduction that does not enlarge easy part proportion and does not reduce positive domain proportion.

2 Reduction from Samplable to the Uniform Distributions

We prove our main theorem explained in Introduction. Below the symbols are the same as those used in the theorem.

2.1 Our contribution: Why can't we simply use the technique of [IL90]?

Before stating the proof, we recall the approach of Impagliazzo and Levin and explain our technical contribution.

We would like to convert the distribution $\mathcal{H}_n$ to the uniform distribution $\mathcal{U}_{\overline{n}}$, where $\overline{n}$ is the size of reduced instances obtained from instances for F of length n. For this purpose, we encode each $x \in L$ of length n by a string y of length reflecting its "weight" $\mathcal{H}_n(x)$. For example, a string x_1 with $\mathcal{H}_n(x_1) = 1/4\ (= 2^{-2})$ is encoded by some y_1 of roughly 2 bit length whereas a string x_2 with $\mathcal{H}_n(x_2) = 2^{-10}$ 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. Let $\mathrm{Hash}(n,k)$ denote a family of pair-wise independent hash functions mapping from $\{0,1\}^n$ to $\{0,1\}^k$. We use randomly chosen hash functions $h_1 \in \mathrm{Hash}(n,2)$ and $h_2 \in \mathrm{Hash}(n,10)$ and 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 strings 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: For using hash functions appropriately, we need to estimate the weight of a given x; more specifically, $k_x = -\log \mathcal{H}_n(x)$. But since this estimation is usually hard, we would select k randomly (even so, the chance of $k = k_x$ is not so small). Then there is some possibility that by some $h' \in \mathrm{Hash}(n,2)$, both heavy input x_1 and light one x_2 are mapped to a short code $y' = h'(x_1) = h'(x_2) \in \{0,1\}^2$. In this case even though $\langle h', y', pad' \rangle$ is solved it may be the case that the solution is for the light input x_2 and it does not help to give a reasonable success probability of solving F. The clever idea of [IL90] is to use another hash function g for avoiding this problem. This hash function g is used to check whether a string x mapped to y indeed has weight $\mathcal{H}_n(x)$ that is large enough for the current choice of hash function h (or equivalently the current choice of k). Note that weight $\mathcal{H}_n(x)$ is essentially the same as the number of random seeds with which the generator yields x. We use the hash function g to check whether there are enough number of such random seeds for x. This technique is called in [GT07] *uniquely decodable random coding.*

This last "unique decodability" of the technique of Impagliazzo and Levin is somewhat limited, and it is weak to guarantee enough uniqueness property for our probability situation. More specifically, the technique of [IL90] can bound the "bad case" probability by $1/c$ for some constant $c > 0$. That is, the probability that an algorithm on input $\langle h', y', pad' \rangle$ gives an undesired solution, e.g., a solution for the light x_2 with $h'(x_2) = y'$, is bounded by $1/c$. This is enough when the algorithm gives a correct answer with probability $1 - 1/\mathrm{poly}(n)$, but this is not sufficient for our situation where we can only assume an algorithm with $1/\mathrm{poly}(n)$ success probability. The main technical contribution of this paper is to solve this problem by using triple-wise independent hash function families for g and by introducing a new analysis for the unique decodability. Also some slightly more careful analysis is needed for taking care of the small success probability, and for this, we introduce some pair-wise independent hash function family with some strong nonshrink property (see below).

2.2 Hash function families

We use hash function families with some reasonable independence and related properties. In general, for any $t \geq 2$, let $\mathrm{Hash}_t(n,k)$ denote a family of t-wise independent hash functions mapping from $\{0,1\}^n$ to $\{0,1\}^k$. That is, the following holds for any $x_1, ..., x_t \in \{0,1\}^n$ such that $x_1 \neq \cdots \neq x_t$ and for any $y_1, ..., y_t \in \{0,1\}^k$:

$$\Pr_{h:\mathrm{Hash}_t(n,k)} [\ h(x_1) = y_1 \wedge h(x_2) = y_2 \wedge \cdots \wedge h(x_t) = y_t\ ] \ = \ (2^{-k})^t.$$

Here by "$h : \mathrm{Hash}_t(n,k)$" we mean to choose a hash function h uniformly at random from $\mathrm{Hash}_t(n,k)$ (see below for more precise meaning).

Although we will use only pair-wise and triple-wise hash function families, it has been known (see, e.g., [KJS01]) that for any constant $t \geq 2$, one can define a hash function family

$\mathrm{Hash}_t(n,k)$ so that each one of the family is representable in $(kt/2)(n+\log k)+O(1)$ bits. We may further assume some polynomial $\ell_{\mathrm{hash}}(\cdot)$ such that for any n and any $k \le n$, hash functions in $\mathrm{Hash}_2(n,k)$ (and resp., $\mathrm{Hash}_3(n,k)$) are uniformly generated from a random seed chosen uniformly at random from $\{0,1\}^{\ell_{\mathrm{hash}}(n)}$, and that given its seed and an input string in $\{0,1\}^n$, the specified hash function value is computable in polynomial-time in n. In the following, we will identify these random seeds with the corresponding hash functions. Thus "$h:\mathrm{Hash}_t(n,k)$" precisely means to select this seed uniformly at random from $\{0,1\}^{\ell_{\mathrm{hash}}(n)}$.

In the following, for a pair-wise independent hash function family, we will also require an additional property, which we call "nonshrink property". In the proof of the following lemma, we show a way to define a new pair-wise independent hash function family $\mathrm{Hash}(\cdot,\cdot)$ with the nonshrink property as stated in the lemma. As we can easily see from its definition, $\mathrm{Hash}(\cdot,\cdot)$ satisfies the other properties mentioned above; precisely speaking, by using a bigger polynomial for $\ell_{\mathrm{hash}}(\cdot)$. (The following lemma might have been known in the literature, but we state the proof in Appendix for the sake of completeness.)

Lemma 2. We can define a pair-wise independent hash function family $\mathrm{Hash}(\cdot,\cdot)$ that has the following property for any sufficiently large n and k, $4 \le k \le n$: For any $X \subseteq \{0,1\}^n$ of size $\ge n2^k$, we have

$$\Pr_{h:\mathrm{Hash}(n,k)}\left[\ |h(X)| \ge (1-2^{-n/24})2^k\ \right] \ \ge\ 1-2^{-n/7}.$$

We first explain the idea of a new hash function family. For simplifying our discussion, we assume here that n is sufficiently large and $n=2^l$ for some l so that $\log n$ is an integer.

Since $|X| \ge n2^k$, consider any partition $\{X_1,...,X_n\}$ of X into sets of almost the same size. Also consider n hash functions $h_1,...,h_n$ chosen independently from $\mathrm{Hash}_2(n,k)$. Then since $|X_i| \approx 2^k$, we may expect that $|h_i(X_i)| \ge 2^k/4$ holds with reasonable probability. In order words, for each $y \in \{0,1\}^k$, the probability that $h_i(x)=y$ for some $x \in X_i$ is not so small. Then since $h_1,...,h_n$ are all independent, we can show that $h_i(x)=y$ for some i and some $x \in X_i$ with probability exponentially close to 1. Now define $h(x)=h_{i(x)}(x)$, where $i(x)$ is the index i such that $x \in X_i$, and we may expect that $h(X)$ occupies the most of its range $\{0,1\}^k$ with probability very close to 1. This is the idea of our hash function family.

For implementing this idea, we need to define the index function $i(\cdot)$ concretely. Here we make use of $h^{(1)} \in \mathrm{Hash}_2(n,l)$ and use its value in $\{0,1\}^l$ as an index. Also we use 2^l pair-wise independent randomly generated hash functions, that is, $h^{(2)}_u \in \mathrm{Hash}_2(n,k)$ for each $u \in \{0,1\}^l$. Then define h by

$$h(x) \ = \ h^{(2)}_u(x), \text{ where } u=h^{(1)}(x).$$

Let $\mathrm{Hash}(n,k)$ denote the set of hash functions obtained in this way. This also defines the way to generate each one in $\mathrm{Hash}(n,k)$ randomly. Clearly, this hash function family is pair-wise independent, and it satisfies all the other properties mentioned before the lemma. Also we can prove that this hash function family satisfies the property of the lemma.

2.3 Detail Argument

Now we state the proof of our main theorem in detail. We begin with recalling notations and introducing some more for our discussion. Throughout this section, we consider sufficiently large n, fix it, and consider only strings of length n for inputs to F. This n is our principle size parameter and the other size parameters are defined based on it.

F : original NP-search problem,	G : NP-search problem reduced from F,
L : set of F's (length n) positive inputs,	M : set of G's (length $\overline{n}$) positive inputs,
x : instance for F (length n),	$\overline{x}$: instance for G (length $\overline{n}$),
$F(x) \;=\; F$'s solution set on x,	$G(\overline{x}) \;=\; G$'s solution set on $\overline{x}$,
$\mathcal{H}_n$: distribution for F,	$\mathcal{U}_{\overline{n}}$: uniform distribution (for G).

Here are some more notations concerning F and $\mathcal{H}$. Note that n' is a new symbol.

H : polynomial-time sampler for $\mathcal{H}$, $\quad n' \stackrel{\text{def}}{=} n^s \;=$ length of random seeds used by $H(1^n)$,
$H(1^n; r) \;=$ length n instance generated by H by using seed $r \in \{0,1\}^{n'}$.

Finally we introduce some polynomial-time computable and invertible one-to-one tupling function, and for any $a_1, ..., a_k$, let $\langle a_1, ..., a_k \rangle$ denote the output string of this function on $a_1, ..., a_k$. For some constant c_{tuple}, we assume that $|\langle a_1, ..., a_k \rangle| = (|a_1| + \cdots + |a_k|)^{c_{\text{tuple}}}$ (when $|a_1| + \cdots + |a_k|$ is sufficiently large) so that the length $|\langle a_1, ..., a_k \rangle|$ is determined by the total length of the input strings $a_1, ..., a_k$. Note that c_{tuple} is one of the universal constants.

Now we define our target problem G based on a given search problem F and a sampler H. An input $\overline{x}$ for G is defined as $\overline{x} = \langle k, h, y, g, pad \rangle$; that is, it consists of five components with the following domains:

G's input is $\overline{x} = \langle k, h, y, g, pad \rangle$,
where $\cdot$ k is from $[n]$, $\quad\cdot$ h is from $\mathrm{Hash}(n, k)$,
$\cdot$ y is from $\{0,1\}^k$, $\quad\cdot$ g is from $\mathrm{Hash}_3(n', n' - (k + 2\log n))$, and
$\cdot$ pad is from $\{0,1\}^\ell$ for some appropriate length ℓ

We may assume that all inputs $\overline{x}$ have the same length. Precisely speaking, k is expressed as a binary string of some fixed length, say, $\lceil \log(n+1) \rceil$, and h and g are seed strings in $\{0,1\}^{\ell_{\text{hash}}(n)}$ and $\{0,1\}^{\ell_{\text{hash}}(n')}$. Since g is the longest in the five components of $\overline{x}$ and $|g| \le \ell_{\text{hash}}(n^s)$, there is some universal constant l_0, with which we may assume that $\overline{n}$ $(= |\overline{x}|) = n^{l_0 \cdot s}$.

We consider the average-case situation where these inputs $\overline{x}$ is given following the uniform distribution $\mathcal{U}_{\overline{n}}$. Here the uniform distribution $\mathcal{U}_{\overline{n}}$ means to choose an instance $\overline{x}$ uniformly at random from $\{0,1\}^{\overline{n}}$. This is in fact equivalent to generate $\overline{x} = \langle k, h, y, g, pad \rangle$ by choosing components k, h, y, g, and pad uniformly at random from each of their domains because the length of each component of $\overline{x}$ is fixed for a given size parameter n and those components are separable from one random string of length $\overline{n}$. Thus, in the following analysis, instead of considering $\mathcal{U}_{\overline{n}}$, we simply argue by assuming that each component is chosen uniformly at random from its domain.

For a given instance $\overline{x} = \langle k, h, y, g, pad \rangle$ of G, the set of solutions $G(\overline{x})$ is defined as follows:

$$
\begin{aligned}
G(\overline{x}) \;=\; \{ \quad & \langle x, r_1, r_2, r_3, w \rangle \;\mid\; x \in \{0,1\}^n,\ r_1, r_2, r_3 \in \{0,1\}^{n'},\ w \in \{0,1\}^m \\
& \text{(a) } h(x) = y, \\
& \text{(b) } w \in F(x) \text{ (i.e., } w \text{ is one of the solution for } F\text{)}, \\
& \text{(c) } H(n; r_1) = H(n; r_2) = H(n; r_3) = x, \text{ and} \\
& \text{(d) } g(r_1) \in 000^* \wedge g(r_2) \in 010^* \wedge g(r_3) \in 100^* \qquad \}.
\end{aligned}
$$

In the following, we will often write $\overline{r}$ for (r_1, r_2, r_3). Clearly the problem of finding *some* solution in $G(\overline{x})$ for a given instance $\overline{x}$ is an NP-search problem. We will show that this search problem has the desired hardness

Now let us prove that G satisfies the theorem. We show that the first requirement (i.e., the statement 4) holds based on the condition 1 of the theorem. That is, the following lemma.

Lemma 3. Suppose that $\Pr_{x:\mathcal{H}_n}[x \in L] \geq n^{-d}$ for some $d \geq 0$. Then for some universal constant $d_0 > 0$, we have

$$\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}[\, G(\overline{x}) \neq \emptyset \,] \;\geq\; (\overline{n})^{-(d+d_0)}.$$

Proof. In order to simplify our presentation, we consider the case that $d = 0$; that is, the case that $\mathcal{H}_n(L) = 1$. The general case can be proven similarly with the same d_0.

For each i, $0 \leq i \leq n'$, define L_i by $L_i = \{x | 2^{-(i+1)} < \mathcal{H}_n(x) \leq 2^{-i}\}$. Noting that $n' = n^s$ and $|L| \leq 2^n$, we have (for sufficiently large n) that

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

Hence, there is some i_0, $0 \leq i_0 \leq n + s\log n$, such that $\mathcal{H}_n(L_{i_0}) \geq 1/2(n + s\log n + 1) > 1/4n$. Note on the other hand that k for the first component of $\overline{x}$ is chosen from $[n]$; thus, letting $k_0 = \min(i_0, n)$, we analyze the probability $\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}[\, G(\overline{x}) \neq \emptyset \,|\, k = k_0 \,]$, i.e., the probability that $\langle k_0, h, y, g, pad\rangle$ has a solution.

First we check that k_0 is not so far from i_0. Note that $|L_{i_0}| \geq 2^{i_0}/4n$ because $\mathcal{H}_n(x) \leq 2^{-i_0}$ for all $x \in L_{i_0}$ and $\mathcal{H}_n(L_{i_0}) \geq 1/4n$. On the other hand, since $|L_{i_0}| \leq 2^n$ (because $L_{i_0} \subset \{0,1\}^n$), we have $2^{i_0}/4n \leq |L_{i_0}| \leq 2^n$; then it follows that $i_0 \leq n + \log n + 2 \leq n + 2\log n$. 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 $|H^{-1}(1^n, x)| \geq 2^{n'-(i_0+1)}$ for all $x \in L_{i_0}$. Now by using this and the bounds $|L_{i_0}| \geq 2^{i_0}/4n$ and $k_0 \leq i_0 \leq k_0 + 2\log n$ derived above, and also by using the independence properties of h and g, we have the following bound.

$$\begin{aligned}
&\Pr_{h,y,g,pad}[\, G(\langle k_0, h, y, g, pad\rangle) \neq \emptyset \,] \\
&\geq \Pr_{h,y,g,pad}[\, \exists x, \overline{r}, w\, [\, \text{(a)}{\sim}\text{(d) holds for } h, y, g, x, \overline{r}, w \,]\,] \\
&\geq \sum_{x \in L_{i_0}} \Pr_{h,y,g,pad}[\, \exists \overline{r}\, [\, \text{(a),(c),(d) holds}^{*1} \text{ for } h, y, g, \overline{r} \text{ on } x \,]\,] \\
&\qquad - \sum_{x \neq x' \in L_{i_0}} \Pr_{h,y,g,pad}[\, (\cdots \text{ on } x) \wedge (\cdots \text{ on } x') \,] \\
&\qquad\qquad\qquad \text{(Note $*1$: (b) is satisfied by considering only } x \in L_{i_0}) \\
&\geq \frac{1}{2} \cdot \sum_{x \in L_{i_0}} \Pr_{h,y,g,pad}[\, \exists \overline{r}\, [\, \text{(a),(c),(d) holds}^{*1} \text{ for } h, y, g, \overline{r} \text{ on } x \,]\,] \\
&\qquad\qquad\qquad \text{(Since we may assume that } \textstyle\sum_{x \in L_{i_0}} \Pr[\cdots] \leq 1/2) \\
&\geq \frac{1}{2} \cdot \sum_{x \in L_{i_0}} \Pr_{h,y}[\, \text{(a) for } h, y, x \,] \cdot \Pr_{g}[\, \exists \overline{r}\, [\, \text{(c),(d) for } g, \overline{r}, x \,]\,] \\
&\geq \frac{|L_{i_0}|}{2} \cdot \sum_{y \in \{0,1\}^{k_0}} \frac{1}{2^{k_0}} \cdot \Pr_{h}[\, \text{(a) for } h, y, x \,] \cdot \sum_{\overline{r}:*2} \Pr_{g}[\, \text{(d) for } g, \overline{r} \,]
\end{aligned}$$

(Note $*2$: $\overline{r} = (r_1, r_2, r_3)$ consists of three different elements of $H^{-1}(1^n, x)$)

$$\begin{aligned}
&\geq \frac{|L_{i_0}|}{2} \cdot \frac{2^{k_0}}{2^{k_0} \cdot 2^{k_0}} \cdot \frac{2^{n'-(i_0+1)} \cdot (2^{n'-(i_0+1)} - 1) \cdot (2^{n'-(i_0+1)} - 2)}{2^{n'-(k_0+2\log n)} \cdot 2^{n'-(k_0+2\log n)} \cdot 2^{n'-(k_0+2\log n)}} \\
&\geq \frac{2^{i_0}}{8n} \cdot \frac{2^{k_0}}{2^{k_0} \cdot 2^{k_0}} \cdot \frac{(2^{n'-(i_0+1)})^3}{2 \cdot (2^{n'-(k_0+2\log n)})^3} \\
&\geq \frac{1}{8n} \cdot \frac{1}{16} \geq \frac{1}{128n}
\end{aligned}$$

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

Next consider the second requirement (i.e., the statement 5) of the theorem. For this we assume an algorithm B solving G with (maybe small but) nonnegligible probability. Although (1) of the theorem states somewhat weak success bound (namely, 1/2) of $B(\overline{x})$, by the standard technique, we may improve this bound to, say, $1 - 2^{-n}$, i.e., the one exponentially close to 1. Thus, for simplifying our discussion, we will argue below from a stronger assumption that some *deterministic* and $(\overline{n})^t$-time bounded algorithm B satisfies

$$\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}[\, B(\overline{x}) \text{ yields some solution} \in G(\overline{x}) \;\mid\; \overline{x} \in M \,] \;\geq\; (\overline{n})^{-e}. \tag{3}$$

An algorithm A for the original NP-search problem F is then defined by using this algorithm B. It is in fact defined in the following simple way.

```
Algorithm: $A$ (input $x$)
1) Choose $k \in [n]$, $h \in \mathrm{Hash}(n,k)$, $g \in \mathrm{Hash}(n', n' - (k + 2\log n))$, and $pad \in \{0,1\}^l$
   uniformly at random;
2) Execute $B(\langle k, h, h(x), g, pad \rangle)$ and output the last component $w$ of the output of $B$.
```

Clearly this algorithm's running time is determined by the time for computing $h(x)$ and the running time of $B(\langle k, h, h(x), g, pad\rangle)$; hence, this can be bounded by $(n^{l_0 \cdot s})^{t + c_0'} \leq n^{c_0 \cdot s \cdot t}$ with some constant $c_0 \stackrel{\text{def}}{=} l_0 \cdot (c_0' + 1)$. Note that this constant c_0 $(= l_0 \cdot (c_0' + 1))$ is a universal constant.

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 4. With some universal constant $e_0 > 0$, the following holds.

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

We prove the lemma by a sequence of claims. First we analyze the performance of the algorithm B, based on the assumption (3), which is restated as follows in terms of n.

Remark on Notations. In the following technical discussion, we assume the condition "$\overline{x} \in M$" omit stating this condition explicitly every time. Also when a random variable follows the uniform distribution on its domain, we simply write, e.g., $\Pr_{\overline{x}}[\cdots]$ as an abbreviation of $\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}[\cdots]$.

$$\Pr_{\overline{x}}[\, B(\overline{x}) \in G(\overline{x})\,] \;\geq\; (\overline{n})^{-e} \;=\; n^{-l_0 \cdot s \cdot e}.$$

Recall that $\Pr_{\overline{x}}[B(\overline{x}) \in G(\overline{x})]$ is $\Pr_{\langle k,y,h,g,pad\rangle}[B(\langle k,y,h,g,pad\rangle) \in G(\langle k,y,h,g,pad\rangle)]$ and that k is chosen from $[n]$. Hence, we can fix some k_0 so that

$$\Pr_{h,y,g,pad}[B(\langle k_0,y,h,g,pad\rangle) \in G(\overline{x})] \geq n^{-l_0\cdot s\cdot e}/(n+1) > n^{-(l_0\cdot s\cdot e+2)}.$$

holds. Here we treat the case where k_0 is small, say, $k_0 < 4$ as a special case; in this case it is easy to show the existence of positive and relatively heavy instances for F that can be solved by A defined above, which is enough to guarantee (1). Thus, in the following we consider the situation where $k_0 \geq 4$. From now on we focus instances of the form $\langle k_0, h, y, g, pad\rangle$, and we will use $\overline{x}_0$, or sometimes $\overline{x}_0\langle h,y\rangle$, $\overline{x}_0\langle h,y,g\rangle$, or $\overline{x}_0\langle h,y,g,pad\rangle$ to denote it.

We use the following variation of the Markov inequality.

Proposition 5. 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 $\gamma = n^{-(l_0\cdot s\cdot e+2)}$ and apply this proposition twice to the above bound. Then we have the following two claims.

Claim 1. We say that h is *good* if $\Pr_{y,g,pad}[B(\overline{x}_0) \in G(\overline{x}_0)] \geq \gamma/2$. The proportion of good h is at least $\gamma/2$. That is,

$$\Pr_{h}\left[\ \Pr_{y,g,pad}[B(\overline{x}_0) \in G(\overline{x}_0)] \geq \frac{\gamma}{2}\ \right] \geq \frac{\gamma}{2}.$$

Claim 2. Consider any good h and fix it. We say that y is *good* (w.r.t. h) if $\Pr_{g,pad}[B(\overline{x}_0) \in G(\overline{x}_0)] \geq \gamma/4$. The proportion of good y is at least $\gamma/4$. That is, we have

$$\Pr_{y}\left[\ \Pr_{g,pad}[B(\overline{x}_0) \in G(\overline{x}_0)] \geq \frac{\gamma}{4}\ \right] \geq \frac{\gamma}{4}.$$

We let $b = 3(l_0\cdot s\cdot e+2)+1$ so that $n^{-b} \leq (\gamma/8)^3$ (for sufficiently large n), and let $a = b+6$. Then we say that a positive instance $x \in L$ is *fat* if its weight $\mathcal{H}_n(x)$ satisfies $\mathcal{H}_n(x) \geq 1/(n^a K_0)$. Note that there are at most $n^a K_0$ fat instances in L ($\subseteq \{0,1\}^n$). Let L_{fat} denote a set consisting of all fat instances in L and some dummy strings[1] so that $|L_{\text{fat}}| = n^a K_0$. Then the following claim holds.

Claim 3.

$$\Pr_{h,y,g}[\ \exists x \in L_{\text{fat}}[h(x)=y] \wedge B(\overline{x}_0\langle h,y,g\rangle) \in G(\overline{x}_0\langle h,y,g\rangle)\ ] \geq \frac{\gamma^3}{128} \geq 4n^{-b}.$$

Proof. The claim is proven by counting all h, y, and g satisfying the condition. We say that a hash function $h \in \text{Hash}(n,k_0)$ *nonshrink* (w.r.t. the set L_{fat}) if $|h(L_{\text{fat}})| \geq K_0(1-\gamma/8)$ holds. Since $|L_{\text{fat}}| = n^a K_0$ with $a \geq 1$, by the nonshirink property of $\text{Hash}(n,k_0)$ (Lemma 2), we have (for sufficiently large n) $|h(L_{\text{fat}})| \geq K_0(1-2^{-n/24}) \geq K_0(1-\gamma/8)$ by at least $1-2^{-n/7} > 1-\gamma/8$

[1]It may be the case that $n^a K_0 \geq 2^n$. Then $L_{\text{fat}} = \{0,1\}^n$; analysis for this case is easier and omitted.

of all h's in $\mathrm{Hash}(n, k_0)$. 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 is $\geq \gamma/2 - \gamma/8 > \gamma/4$.

For each nonshrink and good h, we have at least $K_0(1-\gamma/8)$ y's that have some $x \in L_{\text{fat}}$ such that $y = h(x)$. On the other hand, there are at least $K_0\gamma/4$ good y's, i.e., y's for which $B(\overline{x}_0\langle h, y, g, pad\rangle) \in G(\overline{x}_0\langle h, y, g, pad\rangle)$ holds for at least $\gamma/4$ of all g's and *pad*'s. Hence, the probability that random y, g, and *pad* satisfy both $y = h(x)$ and $B(\overline{x}_0) \in G(\overline{x}_0)$ is at least $(\gamma/4 - \gamma/8) \cdot \gamma/4 = \gamma^2/32$. 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 $B(\overline{x}_0) \in G(\overline{x}_0)$ into *disjoint* subcases by considering the output of B.

$$
\begin{aligned}
4n^{-b} &\leq \Pr_{h,y,g,pad}[\ \exists x \in L_{\text{fat}}[h(x) = y] \wedge B(\overline{x}_0) \in G(\overline{x}_0)\ ] \\
&= \sum_{x'} \Pr_{h,y,g,pad}[\ \exists x \in L_{\text{fat}}[h(x) = y] \wedge B(\overline{x}_0) = (x', \overline{r}, w) \in G(\overline{x}_0)\ ] \\
&= \sum_{x' \in L_{\text{fat}}} \Pr_{h,y,g,pad}[\ \exists x \in L_{\text{fat}}[h(x) = y] \wedge B(\overline{x}_0) = (x', \overline{r}, w) \in G(\overline{x}_0)\ ] \\
&\quad + \sum_{x'' \notin L_{\text{fat}}} \Pr_{h,y,g,pad}[\ \exists x \in L_{\text{fat}}[h(x) = y] \wedge B(\overline{x}_0) = (x'', \overline{r}, w) \in G(\overline{x}_0)\ ] \\
&\leq \sum_{x' \in L_{\text{fat}}} \Pr_{h,y,g,pad}[\ B(\overline{x}_0) = (x', \overline{r}, w) \in G(\overline{x}_0)\ ] \qquad (4) \\
&\quad + \sum_{x'' \notin L_{\text{fat}}} \sum_{x \in L_{\text{fat}}} \Pr_{h,y,g,pad}[\ h(x) = y \wedge B(\overline{x}_0) = (x'', \overline{r}, w) \in G(\overline{x}_0)\ ]. \qquad (5)
\end{aligned}
$$

Consider the last two terms, i.e., (4) and (5). Noting that $h(x') = y$ is a part of the condition $(x', \overline{r}, w) \in G(\overline{x}_0)$, we can restate (4) as follows.

$$
\begin{aligned}
(4) &= \sum_{x' \in L_{\text{fat}}} \Pr_{h,y,g,pad}[\ h(x') = y \wedge B(\overline{x}_0) = (x', \overline{r}, w) \in G(\overline{x}_0)\ ] \\
&= \sum_{x' \in L_{\text{fat}}} \Pr_{h,y,g,pad}[\ B(\overline{x}_0) = (x', \overline{r}, w) \in G(\overline{x}_0) \ \mid\ h(x') = y\ ] \cdot \Pr_{h,y}[\,h(x') = y\,] \\
&= \sum_{x' \in L_{\text{fat}}} \Pr_{h,y,g,pad}[\ B(\overline{x}_0\langle h, y\rangle) = (x', \overline{r}, w) \in G(\overline{x}_0\langle h, y\rangle) \ \mid\ h(x') = y\ ] \cdot \frac{1}{K_0} \\
&= \frac{1}{K_0} \cdot \sum_{x' \in L_{\text{fat}}} \Pr_{h,y,g,pad}[\ B(\overline{x}_0\langle h, h(x')\rangle) = (x', \overline{r}, w) \in G(\overline{x}_0\langle h, h(x')\rangle)\ ] \\
&= \frac{1}{K_0} \cdot \sum_{x' \in L_{\text{fat}}} \Pr_{h,g,pad}[\ B(\overline{x}_0\langle h, h(x')\rangle) = (x', \overline{r}, w) \in G(\overline{x}_0\langle h, h(x')\rangle)\ ]
\end{aligned}
$$

Intuitively, this is the total success probability of our procedure A. On the other hand, the term (5) bounds the probability that B's answer does not help us for solving $x' \in L_{\text{fat}}$. Our new technique of using triple-wise independent hash functions and its analysis, which is different from the one in [IL90], are for bounding this probability small. More specifically, we can bound it as the following claim.

Claim 4. For any $x \in L_{\text{fat}}$, we have

$$
\sum_{x'' \notin L_{\text{fat}}} \Pr_{h,y,g,pad}[\ h(x) = y \wedge B(\overline{x}_0\langle h, y, g\rangle) = (x'', \overline{r}, w) \in G(\overline{x}_0\langle h, y, g\rangle)\ ] \ \leq\ \frac{n^6}{n^{2a} K_0}.
$$

Thus, since $a = b + 6$ and $|L_{\text{fat}}| = n^a K_0$, we have

$$(5) \;\le\; \sum_{x \in L_{\text{fat}}} \frac{n^6}{n^{2a} K_0} \;=\; n^{-b}.$$

Proof. Fix any $x \in L_{\text{fat}}$. The claim is shown by the following analysis.

$$\begin{aligned}
&\sum_{x'' \notin L_{\text{fat}}} \Pr_{h,y,g,pad}[\; h(x) = y \,\wedge\, B(\overline{x}_0\langle h,y,g\rangle) = (x'',\overline{r},w) \in G(\overline{x}_0\langle h,y,g\rangle) \;] \\
&\le \sum_{x'' \notin L_{\text{fat}}} \sum_{r_1:*1} \sum_{r_2:*1} \sum_{r_3:*1} \Pr_{h,y,g,pad}[\; h(x) = h(x'') = y \,\wedge\, \text{(d) holds for } g \text{ and } \overline{r} \;] \\
&\qquad\qquad\qquad\qquad (\text{Note } *1\text{: each } r_i \text{ is chosen so that } H(1^n; r_i) = x'') \\
&\le \sum_{r_1:*2} \sum_{r_2:*3} \sum_{r_3:*3} \Pr_{h,y,g,pad}[\; h(x) = h(x'') = y \,\wedge\, \text{(d) holds for } g \text{ and } \overline{r} \;] \\
&\quad \left(\begin{array}{ll} \text{Note} & *2\text{: } r_1 \text{ is chosen so that } x'' \overset{\text{def}}{=} H(1^n; r_1) \notin L_{\text{fat}} \\ & *3\text{: } r_i \text{ is chosen so that } H(1^n; r_i) = x'' \end{array} \right) \\
&\le 2^{n'} \cdot \frac{2^{n'-k_0}}{n^a} \cdot \frac{2^{n'-k_0}}{n^a} \cdot \frac{1}{K_0^2} \cdot \left(\frac{1}{2^{n'-(k_0+2\log n)}} \right)^3 \;=\; \frac{n^6}{n^{2a} K_0}.
\end{aligned}$$

Here we use the fact that $x'' \notin L_{\text{fat}}$ implies that $\mathcal{H}_n(x'') < 1/(n^a K_0)$; in other words, the number of r such that $H(1^n; r) = x''$ is less than $2^{n'-k_0}/n^a$. □

From the above claim and the restatement of (4), we have

$$\frac{1}{n^a K_0} \cdot \sum_{x' \in L_{\text{fat}}} \Pr_{h,g,pad}[\; B(\overline{x}_0\langle h, h(x')\rangle) \in G(\overline{x}_0\langle h, h(x')\rangle) \;] \;\ge\; 3n^{-(a+b)} \;\ge\; 2n^{-(a+b)}. \qquad (6)$$

From this we now show that there are enough x''s for which A succeeds with our desired probability. We say that x is A-*good* if

$$\Pr_{h,g,pad}[\; B(\langle k_0, h, h(x), g, pad\rangle) \in G(\langle k_0, h, h(x), g, pad\rangle) \;] \;\ge\; n^{-(a+b)}$$

Recall that $a + b = 2b + 6 = 6(l_0 \cdot s \cdot e + 2) + 8$. Hence, we may choose the universal constant e_0 of the lemma large enough so that $n^{-(a+b)} > n^{-e_0 \cdot s \cdot e}$ holds. Thus, A-good x's are those for which $A(x)$ has the desired success probability. Therefore, the lemma is proven by the following claim.

Claim 5.

$$\Pr_{x:\mathcal{H}_n}[\; x \text{ is } A\text{-good} \;] \;\ge\; n^{-(a+b)} \;>\; n^{-e_0 \cdot s \cdot e}.$$

Proof. By applying Proposition 5 to the bound (6), we have

$$\left| \left\{ x \;\middle|\; \Pr_{h,g,pad}[B(\overline{x}_0\langle h, h(x)\rangle) = (x,\overline{r},w) \in G(\overline{x}_0)] \ge n^{-(a+b)} \right\} \right| \;\ge\; \frac{n^a K_0}{n^{a+b}}.$$

Recall that $\overline{x}_0\langle h, h(x)\rangle$ is the abbreviation of $\langle k_0, h, h(x), g, pad\rangle$; hence, the above means that the number of A-good x's is at least $n^a K_0 \cdot n^{-(a+b)}$.

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

$$\Pr_r[\,H(1^n;r)=x\,] \geq \frac{1}{n^a K_0} = \frac{2^{n'-k_0}}{n^a \cdot 2^{n'}}.$$

This is because if otherwise, we have $|\{r : H(1^n;r)=x\}| < 2^{n'-k_0}/n^a$, and hence

$$\begin{aligned}&\Pr_g[(*)\,\exists\overline{r}\,[\,H(1^n;r_1)=H(1^n;r_2)=H(1^n;r_3)=x \;\wedge\; \text{(d) holds for } \overline{r} \text{ and } g\,]\,]\\ &< \left(\frac{2^{n'-k_0}}{n^a}\right)^3 \cdot \left(\frac{1}{2^{n'-(k_0+2\log n)}}\right)^3 \;=\; \frac{n^6}{n^{3a}} \;<\; n^{-(a+b)},\end{aligned}$$

but then since $(*)$ is a part of the conditions for solutions of x, the probability that $A(x)$ gives any solution must be less than this bound, contradicting the assumption that x is A-good.

Now we know that there are at least $n^a K_0 \cdot n^{-(a+b)}$ good x's and that each of them is fat, i.e., $\mathcal{H}_n(x) \geq 1/(n^a K_0)$. This proves that the probability of good x under the distribution $\mathcal{H}_n$ is at least $n^{-(a+b)}$. □

3 Concluding Remarks

We show a strong hardness preserving reduction for distributional NP-search problems from any polynomial-time samplable distribution to the uniform distribution. We needed to consider NP-search problems because our current reduction may create nonnegligible fraction of "no" instances and this makes the problem easy as a decision problem; small but still nonnegligible correct probability can be achieved even by the trivial algorithm that always yields "no" answer.

Although not yet certain, there may be some approach for overcoming this obstacle. First we remark that the proportion of a positive domain M of our constructed uniformly hard problem G can be made very close to 1. The hard problem G shown in Theorem 1 has a positive domain M that occupies at least $1/\mathrm{poly}(\overline{n})$ of $\{0,1\}^{\overline{n}}$ for each sufficiently large $\overline{n}$; but we can in fact modify it to some G' with a positive domain M' that is close to $\{0,1\}^{n'}$. More precisely, we can prove the following a bit stronger version of Theorem 1.

Theorem 6. *Let G and M be an NP-search problem and its positive domain defined in Theorem 1 from the original NP-search problem F and a distribution $\mathcal{H}$. In particular, recall that G and M satisfying the conditions 4 and 5 of the theorem with constants d and e. Then we can define some NP-search problem G' and its positive domain M' satisfying the following conditions with some universal constant $c_1 > 0$ and some constant $c \geq 1$ that depends on d and e.*

1. $\Pr_{x':\mathcal{U}_{n'}}[\,x' \in M'\,] \;\geq\; 1 - 2^{-c_1 n'}$, and
2. if some polynomial-time randomized search algorithm B' for G' achieves

$$\Pr_{x':\mathcal{U}_{n'}}\left[\;\Pr_{B'}[B'(x') \text{ yields some solution} \in G'(x')\,] \;\geq\; 1/2\;\right] \;\geq\; (n')^{-e/c_2}, \tag{7}$$

then by using B' one can define some polynomial-time randomized search algorithm B that solves G as shown in the following sense of (1), which is restated as follows:

$$\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}\left[\;\Pr_{B}[B(\overline{x}) \text{ yields some solution} \in G(\overline{x})\,] \;\geq\; 1/2 \;\mid\; \overline{x} \in M\;\right] \;\geq\; (\overline{n})^{-e}.$$

Thus, by Theorem 1, we may construct some polynomial-time randomized algorithm A that solves F with nonnegligible probability under $\mathcal{H}$.

Proof. The idea is to apply a (random) hash function with the nonshrink property to positive instances $\overline{x}$ of G to define instances x' of G'; that is, the set M' of positive instances of G' is defined by $M' = \{(h(\overline{x}), h) \mid \overline{x} \in M \,\text{and}\, h \in \text{Hash}(|\overline{x}|, k(|\overline{x}|))\}$. A solution of a positive instance $x' = (h(\overline{x}), h)$ is simply $\overline{x}$ itself and a solution of $\overline{x}$ w.r.t. G.

Fix $\overline{n}$, the length of G's instances, and show the above idea works with some appropriate choice of parameters. First fix $k = k(\overline{n}) = \overline{n} - (d_0 + d)\log\overline{n} - 1$, and use $\text{Hash}(\overline{n}, k)$ as the hash function family with the nonshrink property. Then G' is defined as above and let $M' = \{(h(\overline{x}), h) \mid \overline{x} \in M \,\text{and}\, h \in \text{Hash}(\overline{n}, k)\}$ (which is a slice of the actual M' corresponding to G's instances of size $\overline{n}$). Let $\ell = \ell_{\text{hash}}(\overline{n})$ be the length of each hash function of $\text{Hash}(\overline{n}, k)$. Let n' be the length of strings of the form $(h(\overline{x}), h)$; we may assume that $\overline{n} \le n' \le \overline{n}^{O(1)}$.

Note first that

$$|M| \;\ge\; \frac{2^{\overline{n}}}{\overline{n}^{d_0+d}} \;=\; \overline{n}2^{\overline{n}-(d_0+d)\log\overline{n}-1} \;=\; \overline{n}2^k.$$

Thus, by the nonshrink property of Lemma 2, we have

$$|M'| \;\ge\; (1 - 2^{-\overline{n}/24})(1 - 2^{-\overline{n}/7})2^k 2^\ell,$$

from which the condition 1 of the theorem follows with some universal parameter $c_1 > 0$.

For the condition 2 of the theorem, assume some polynomial-time randomized algorithm B' that satisfies (7). An instance x' is called *G'-easy* if $B'(x')$ yields a solution with prob. $> 1/2$. We may assume that for such easy instances, B' yields a solution with prob. very close to 1; this can be achieved by running the original B', say, n' times. Now our search algorithm B executes the following steps on a given input $\overline{x}$ of length $\overline{n}$: generate a random hash function $h \in Hash(\overline{n}, k)$, execute B' on $x' = (h(\overline{x}), h)$, and output its (the part of its answer) as a solution of $\overline{x}$ (if it is correct).

We analyze this B's success probability; that is, the proportion of instances $\overline{x}$ for which B yields a solution (with a certain probability). Specifically, we call $\overline{x} \in \{0,1\}^{\overline{n}}$ *G-easy* if $x' = (h(\overline{x}), h)$ is G'-easy for $(n')^{-e/c}/2$ of h in $\text{Hash}(\overline{n}, k)$. Clearly, for such G-easy instances, our defined B yields a solution with prob. $> (n')^{-e/c}/2$. Hence, for the proof[2], it suffices to show that there are enough number of G-easy instances. Let us begin with counting the number of G'-easy instances x'. From our assumption (7) and the fact that $n' \ge k + \ell$, we have

$$\#\text{ of } G'\text{-easy instances} \;\ge\; (n')^{-e/c} \cdot 2^{n'} \;\ge\; (n')^{-e/c} \cdot 2^k \cdot 2^\ell \;=\; \alpha \cdot K \cdot L,$$

where $\alpha = (n')^{-e/c}$, $K = 2^k$, and $L = 2^\ell$. Then by an argument for proving Proposition 5, we can show estimate # of $\overline{x}$ such that $(h(\overline{x}), h)$ is G'-easy for at least $(\alpha/2) \cdot L$ many h's of $\text{Hash}(\overline{n}, k)$; this is the number of G-easy instances. Hence, we have

$$\#\text{ of } G\text{-easy instances} \;\ge\; \frac{\alpha}{2} \cdot K.$$

This is in proportion estimated as follows for some sufficiently large c.

$$\Pr_{\overline{x}:\mathcal{U}_{\overline{n}}}[\,\overline{x} \text{ is } G\text{-easy}\,] \;\ge\; \frac{\alpha K/2}{2^{\overline{n}}} \;=\; \frac{(n')^{-e/c} \cdot 2^{\overline{n}-(d_0+d)\log n-2}}{2^{\overline{n}}} \;\ge\; (\overline{n})^{-e/c'-d_0-d-2} \;\ge\; (\overline{n})^{-e}.$$

[2]Precisely speaking, the condition 2 requires the success prob. $> 1/2$, which can be achieved by running the current B for enough number of times.

□

From this theorem, we have a search problem that is solvable and hard on almost all instances (provided the original F has enough hardness under the P-samplable distribution $\mathcal{H}$). Then by applying the combination of the isolation technique and some list decoding as Gutfreund did in [Gu06], we may be able to define some hard decision problem from this NP-search problem. We would like to leave this investigation as our future work.

References

[BT06a] A. Bogdanov and L. Trevisan, On worst-case to average-case reductions for NP problems, *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[BT06b] A. Bogdanov and L. Trevisan, Average-case complexity, *Foundation and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[GS89] S. Goldwasser and M. Sipser, Private coins versus public coins in interactive proof system, *Advances in Computing Research, Vol. 5: Randomness and Computation*, JAI Press, 73–90, 1989.

[Gu06] D. Gutfreund, Worst-case vs. average-case complexity in the polynomial-time hierarchy, in *Proc. RANDOM 2006*, LNCS 4110, 386–397, 2006.

[GT07] D. Gutfreund and A. Ta-Shma, Worst-case to average-case reductions revisited, in *Proc. RANDOM 2007*, LNCS 4627, 569–583, 2007.

[Im95] R. Impagliazzo, Hard-core distributions for somewhat hard problems, in *Proc. FOCS '95*, 538–545, 1995.

[IL90] R. Impagliazzo and L. Levin, No better ways to generate hard NP instances than picking uniformly at random, in *Proc. FOCS '90*, 812–821, 1990.

[KJS01] K. Kurosawa, T. Johansson, and D.R. Stinson, Almost k-wise independent sample spaces and their cryptologic applications, *Journal of Cryptology*, 14(4): 231–253, 2001.

[Yao82] A. Yao, Theory and applications of trapdoor functions (extended abstract), in *Proc. FOCS '82*, 80–91, 1982.

4 Appendix: Proof of Lemma 2

Here we prove Lemma 2, that is, the following nonshrink property of our hash function family $\mathrm{Hash}(n,k)$: For any $X \subset \{0,1\}^n$ of size $\geq nK$ (where $K = 2^k$), we have

$$\Pr_{h:\mathrm{Hash}(n,k)} \left[\; |h(X)| \geq (1 - 2^{-n/24})K \; \right] \;\geq\; 1 - 2^{-n/7}.$$

We prove this bound by a sequence of claims. Here we consider that, whenever necessary, $h^{(1)}$ and $h_u^{(2)}$ are chosen uniformly at random from $\mathrm{Hash}_2(n,l)$ and $\mathrm{Hash}_2(n,k)$ respectively. Also we fix the domain of u and y to $\{0,1\}^l$ and $\{0,1\}^k$ respectively.

First, for each $u \in \{0,1\}^l$, we define $X_u = \{x \in X \mid h^{(1)}(x) = u\}$. Then we have the following claim.

Claim 6. . For every $u \in \{0,1\}^l$,

$$\Pr_{h^{(1)}}[\,|X_u| \geq K/2\,] \;\geq\; 1/2.$$

Proof. Fix any $u \in \{0,1\}^l$. For any x, define a random variable $I_{u,x} = 1$ if $h^{(1)}(x) = u$ and 0 otherwise, and let $I_u = \sum_{x\in X} I_{u,x}$. Then we have $|X_u| = I_u$, and it is easy to see that $\mathrm{E}[I_u] = K$. On the other hand, by the pair-wise independence property of $\mathrm{Hash}_2(n,l)$, we can show that $\mathrm{Var}[I_u] \leq K$. Thus, the bound of the claim holds by the Chebyshev bound. □

For each u and y, we define a random variable $J_{y,u}$ as follows:

$$J_{y,u} \;=\; \begin{cases} 1, & \exists x \in X_u[\,h_u^{(2)}(x) = y\,], \\ 0, & \text{otherwise.} \end{cases}$$

Claim 7. . For every u and y,

$$\Pr_{h^{(1)},h_u^{(2)}}[\,J_{y,u} = 1 \mid |X_u| \geq K/2\,] \;\geq\; 1/4.$$

Proof. Fix any u and y. In this proof, we assume that $|X_u| \geq K/2$. Consider any subset X'_u of X_u such that $|X'_u| = K/2$, and fix it. Define $J'_{y,u}$ as $J_{y,u}$ by using X'_u instead of X_u. Then clearly, $J'_{y,u} = 1$ implies $J_{y,u} = 1$. Here we discuss the probability that $J'_{y,u} = 1$. (Note that since X'_u is fixed, the event $J'_{y,u} = 1$ is determined only by $h_u^{(2)}$, independent from the choice of $h^{(1)}$; thus, probabilities we analyze here are on the choice of $h_u^{(2)}$.)

Consider the following sets $U_{y,u}$ and $\overline{U}_{y,u}$.

$$\begin{aligned} U_{y,u} &= \{\,(h_u^{(2)},x) \mid x \in X'_u \wedge h_u^{(2)}(x) = y \wedge \neg\exists x' \in X'_u[h_u^{(2)}(x) = y]\,\}, \\ \overline{U}_{y,u} &= \{\,(h_u^{(2)},x) \mid x \in X'_u \wedge h_u^{(2)}(x) = y \wedge \exists x' \in X'_u[h_u^{(2)}(x) = y]\,\}. \end{aligned}$$

Note that if $h_u^{(2)}$ belongs to $U_{y,u}$ with some $x \in X'_u$, then we have $J'_{y,u} = 1$ with this $h_u^{(2)}$, which is witnessed uniquely by this x. Hence, we have

$$\Pr[J'_{y,u} = 1] \;\geq\; \frac{|U_{y,u}|}{|\mathrm{Hash}_2(n,k)|}. \tag{8}$$

Thus, we analyze the above ratio.

First note that $U_{y,u} \cup \overline{U}_{y,u} = \{(h_u^{(2)},x) \mid h_u^{(2)}(x) = y\}$, and the size of this set is exactly

$$\{(h_u^{(2)},x) \mid h_u^{(2)}(x) = y\} \;=\; \sum_{x\in X'_u} |\{h_u^{(2)} \mid h_u^{(2)}(x) = y\}| \;=\; \frac{K}{2}\cdot\frac{|\mathrm{Hash}_2(n,k)|}{K} \;=\; \frac{|\mathrm{Hash}_2(n,k)|}{2}.$$

Hence, $|U_{y,u} \cup \overline{U}_{y,u}|/|\mathrm{Hash}_2(n,k)| = 1/2$. Thus, for giving a lower bound for $|U_{y,u}|/|\mathrm{Hash}_2(n,k)|$, we analyze an upper bound of $|\overline{U}_{y,u}|/|\mathrm{Hash}_2(n,k)|$. Below we fix the domain of x variables such as x, x', x_1, and x_2 to X'_u, and when, e.g., x is used as a random variable, we always assume that it is chosen from X'_u uniformly at random. First we derive the following bound by using the pair-wise independence of $\mathrm{Hash}_2(n,k)$.

$$\begin{aligned}
&\Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) = h_u^{(2)}(x_2)\,] \\
&= \Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) \wedge x_1 = x_2\,] + \Pr_{h_u^{(2)}}[\, y = h_u^{(2)}(x_1) = h_u^{(2)}(x_2) \mid x_1 \neq x_2\,] \cdot \Pr_{x_1,x_2}[\, x_1 \neq x_2\,] \\
&\leq \Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) \wedge x_1 = x_2\,] + \Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) \wedge y = h_u^{(2)}(x_2) \mid x_1 \neq x_2\,] \\
&\leq \Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) \wedge x_1 = x_2\,] + \frac{1}{K^2}.
\end{aligned}$$

On the other hand, the following bound also holds.

$$\begin{aligned}
&\Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) = h_u^{(2)}(x_2)\,] \\
&\geq \Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) \wedge x_1 = x_2\,] + \Pr_{x_1,x_2,h_u^{(2)}}[\,(h_u^{(2)}, x_1) \in \overline{U}_{y,u} \wedge (\, x_2 \in X'_u - \{x_1\} \wedge h_u^{(2)}(x_2) = y\,)\,] \\
&\geq \Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) \wedge x_1 = x_2\,] + \frac{|\overline{U}_{y,u}|}{|X'_u| \cdot |\mathrm{Hash}_2(n,k)|} \cdot \frac{1}{|X'_u|} \\
&= \Pr_{x_1,x_2,h_u^{(2)}}[\, y = h_u^{(2)}(x_1) \wedge x_1 = x_2\,] + \frac{|\overline{U}_{y,u}|}{|\mathrm{Hash}_2(n,k)|} \cdot \frac{4}{K^2}.
\end{aligned}$$

Then from these bounds we have $|\overline{U}_{y,u}|/|\mathrm{Hash}_2(n,k)| \leq 1/4$, from which (and the above discussion) it follows

$$\Pr[J'_{y,u} = 1] \;\geq\; \frac{|U_{y,u}|}{|\mathrm{Hash}_2(n,k)|} \;\geq\; \frac{1}{2} - \frac{1}{4} \;=\; \frac{1}{4},$$

and the bound of the lemma follows. □

Finally for each $y \in \{0,1\}^k$, we define the following random variable J_y.

$$J_y \;=\; \begin{cases} 1, & \bigvee_{u \in \{0,1\}^l} (J_{y,u} = 1) \wedge (|X_u| \geq K/2), \\ 0, & \text{otherwise,} \end{cases}$$

Then clearly the event $J_y = 1$ implies that (w.r.t. $h^{(1)}$ and $h_u^{(2)}$) there exists some $x \in X$ such that $h(x) = y$; hence, $J \stackrel{\text{def}}{=} \sum_{y \in \{0,1\}^k} J_y$ is a lower bound for $|h(X)|$. Thus, the bound of Lemma 2 follows from the following claim.

Claim 8. .

$$\Pr_{h:\mathrm{Hash}(n,k)}[\, J \geq (1 - 2^{-n/24})K\,] \;\geq\; 1 - 2^{-n/7},$$

Proof. Consider any y. From the previous two claims, for any $u \in \{0,1\}^l$, we have $\Pr_{h^{(1)},h_u^{(2)}}[(J_{y,u} = 1) \wedge (|X_u| \geq K/2)] \geq 1/8$. Then since functions $h_u^{(2)}$, $u \in \{0,1\}^l$, are totally independent, we have

$$\Pr_{h^{(1)},\{h_u^{(2)}\}_{u \in \{0,1\}^l}}\left[\; \bigwedge_{u \in \{0,1\}^l} (J_{y,u} \neq 1) \vee (|X_u| < K/2) \;\right] \;\leq\; \left(1 - \frac{1}{8}\right)^{2^l} \;=\; \left(\frac{7}{8}\right)^{n} \;\leq\; 2^{-n/4}.$$

That is, $\Pr_h[J_y = 1] \geq 1 - 2^{-n/4}$.

Now consider the distribution of $J = \sum_{y\in\{0,1\}^k} J_y$. For its expectation, from the above, we have $\mathrm{E}_h[J] = (1 - 2^{-n/4})K$. On the other hand, its variance is bounded as follows.

$$\begin{aligned}
\mathrm{Var}_h[J] &= \mathrm{E}_h\left[\left(\sum\nolimits_{y\in\{0,1\}^k} J_y\right)^2\right] \;-\; \mathrm{E}_h[J]^2 \\
&= \mathrm{E}_h\left[\sum\nolimits_y J_y^2\right] + \sum\nolimits_{y,y'} \mathrm{E}_h[J_y \cdot J_{y'}] - \mathrm{E}_h[J]^2 \\
&\leq \mathrm{E}_h[J] + K(K-1) - \mathrm{E}_h[J]^2 \;\leq\; K + K^2 - K - (1 - 2^{-n/4})^2 K^2 \;\leq\; 2 \cdot 2^{-n/4} K^2.
\end{aligned}$$

Then by the Chebyshev bound, we have

$$\Pr_h\left[J \;<\; \mathrm{E}_h[J] - \frac{0.9 \cdot 2^{n/12}}{\sqrt{2}} \cdot \frac{\sqrt{2}}{2^{n/8}} \cdot K\right] \;\leq\; \frac{2}{0.81 \cdot 2^{n/6}} \;\leq\; 2^{-n/7},$$

which is equivalent to

$$\Pr_h\left[J \;<\; \left(1 - 2^{-n/4} - 0.9 \cdot 2^{-n/24}\right) K\right] \;\leq\; 2^{-n/7},$$

which proves the claim when n is sufficiently. □