On fast heuristic non-deterministic algorithms and short heuristic proofs

Dmitry Itsykson*
Steklov Institute of Mathematics at St. Petersburg
27 Fontanka, St.Petersburg, 191023, Russia
dmitrits@pdmi.ras.ru

Dmitry Sokolov*
Steklov Institute of Mathematics at St. Petersburg
27 Fontanka, St.Petersburg, 191023, Russia
sokolov.dmt@gmail.com

October 24, 2013

Abstract

In this paper we study heuristic proof systems and heuristic non-deterministic algorithms. We give an example of a language Y and a polynomial-time samplable distribution D such that the distributional problem (Y, D) belongs to the complexity class HeurNP but $Y \notin \mathrm{NP}$ if $\mathrm{NP} \neq \mathrm{coNP}$, and $(Y, D) \notin \mathrm{HeurBPP}$ if $(\mathrm{NP}, \mathrm{PSamp}) \not\subseteq \mathrm{HeurBPP}$.

For a language L and a polynomial q we define the language $pad_q(L)$ composed of pairs (x, r) where x is an element of L and r is an arbitrary binary string of length at least $q(|x|)$. If $D = \{D_n\}_{n=1}^{\infty}$ is an ensemble of distributions on strings, let $D \times U^q$ be a distribution on pairs (x, r), where x is distributed according to D_n and r is uniformly distributed on strings of length $q(n)$. We show that for every language L in AM there is a polynomial q such that for every distribution D concentrated on the complement of L the distributional problem $(pad_q(L), D \times U^q)$ has a polynomially bounded heuristic proof system. Since graph non-isomorphism (`GNI`) is in AM, the above result is applicable to `GNI`.

1 Introduction

1.1 Non-deterministic computations and proof systems

A proof system for a language L is a polynomial-time algorithm $\Pi(x, w)$ such that 1) for all $x \in L$ there exists a string w such that $\Pi(x, w) = 1$ and 2) for all $x \notin L$ and all strings

*The work is partially supported by the Ministry of education and science of Russian Federation, project 8216, the president grants MK-4108.2012.1, by RFBR grants 12-01-31239 mol_a and by RAS Program for Fundamental Research.

w, $\Pi(x, w) = 0$. Informally speaking, the algorithm $\Pi(x, w)$ verifies that w is a proof of the fact $x \in L$. A proof system Π for language L is called polynomially bounded if for every $x \in L$ there exists w of size $poly(|x|)$ such that $\Pi(x, w) = 1$. The set of languages with polynomially bounded proof systems is precisely the class NP. The language TAUT of all propositional tautologies is coNP-complete, hence it has polynomially bounded proof system if and only if NP = coNP. Thus if NP $\neq$ coNP, then every particular proof system for TAUT has a family of hard tautologies that requires superpolynomial proofs. The main activity in the proof complexity theory is proving lower bounds for particular proof systems for TAUT. The program is to prove lower bounds for more and more powerful proof systems and finally to prove that NP $\neq$ coNP. However, to prove NP $\neq$ coNP we have to prove lower bounds for all proof systems. A proof system $\hat{\Pi}$ for language L is p-optimal if for every other proof system Π for L there exists a polynomial-time algorithm that translate Π-proofs to $\hat{\Pi}$-proofs. The existence of a p-optimal proof system is a major open question. If there exists a p-optimal proof system for TAUT, then proving NP $\neq$ coNP reduces to proving a lower bound in one proof system.

An acceptor for language L is an algorithm that stops on the language and does not stop on the complement of the language; an optimal acceptor stops in the minimal (up to a polynomial) possible time on every element of the language. Krajíček and Pudlak in [KP89] show that the existence of an optimal proof system for the language of propositional tautologies TAUT is equivalent to the existence of an optimal acceptor for TAUT. Messner [Mes99] in 1999 extended this result for every paddable language.

1.2 Heuristic computations

Heuristic algorithms solve problems not on all but on almost all inputs according to some distribution. Heuristic computations play a major role in the average-case complexity, but the interest to study them also comes from the structural complexity theory. A number of statements about structures of complexity classes (the time hierarchy theorem, the existence of complete problems, optimal algorithms) are proved in heuristic settings and remain open in the classical sense. Among such results we should note the following: the existence of a complete public-key cryptosystem by Harnik et al [HKN$^+$05] (such a cryptosystem exists if the decryption algorithm is allowed to err on a small fraction of inputs); the time hierarchy theorem for randomized heuristic computations with bounded error proved by Fortnow and Santhanam [FS04] and simplified by Pervyshev [Per07] (for classical randomized algorithms with bounded error the time hierarchy theorem is an open question); the existence of a complete problem in the heuristic analogue of the complexity class BPP by Itsykson [Its10] (the existence of a complete problem in BPP is an open question).

1.3 Heuristic proofs

A distributional problem is a pair of a language L and an ensemble of distributions D. A heuristic proof system [HINS13] for a distributional problem (L, D) is an algorithm $\Pi(x, w, \delta; n)$ such that the following properties are satisfied:

1. For every $x \in L$ there exists $w \in \{0, 1\}^*$ such that $\Pi[x, w, \delta; n] = 1$;

2. $\Pr_{x \leftarrow D_n}[x \notin L \wedge \exists w\ \Pi[x, w, \delta; n] = 1] < \delta$.

3. The running time of Π is bounded by a polynomial in $\frac{n+|w|}{\delta}$.

Informally speaking, the algorithm Π checks that w is a proof of $x \in L$. The number δ is an error parameter: the smaller δ, the fewer incorrect statements of the type $x \in L$ are provable in Π. We say that a system Π is polynomially bounded on $A \subseteq L$ if for every $x \in A$ and $\delta \in (0, 1)$ there exists $w \in \{0, 1\}^*$ such that $|w| < poly(n/\delta)$ and $\Pi[x, w, \delta; n] = 1$.

Heuristic proof systems are related to the complexity class HeurNP that consists of distributional problems (L, D) such that the language L is solved in polynomial in $\frac{|x|}{\delta}$ time by a non-deterministic Turing machine $M(x, \delta)$ on all inputs except a fraction δ (according to D). The machine M may err both on elements of the language L and on elements of the complement of L (in contrast to heuristic proof systems that may err only on the complement of L). If a distributional problem (L, D) has a polynomially bounded heuristic proof system, then $(L, D) \in$ HeurNP. And if a language L is recursively enumerable (i.e. L has an acceptor $\mathcal{L}$) and $(L, D) \in$ HeurNP, then L has the following heuristic proof system Π_M. $\Pi_M(x, w, \delta; n)$ executes two processes in parallel. In the first process the machine $M(x, \delta)$ runs using the string w as a non-deterministic witness. In the second process the algorithm $\mathcal{L}$ runs for at most $|w|$ steps. $\Pi(x, w, \delta; n)$ accepts if at least one of these parallel processes accepts. The proof system Π_M is almost polynomially bounded: all elements of L except a set of the D_n-measure δ have proofs of length $poly(n/\delta)$.

The paper [HIMS12] considers distributional proving problems that are the special case of distributional problems in which the distribution is concentrated on the complement of the language. The paper [HIMS12] shows that for every recursively enumerable language L and every polynomial-time samplable distribution D concentrated on the complement of L there exists an optimal randomized heuristic acceptor for (L, D). A heuristic acceptor stops on every element of the language and also stops on a small fraction of inputs according to D from the complement of L. The corollary of this result is the existence of an optimal weakly automatizable randomized heuristic proof system. However, the proof of the equivalence between the existence of an optimal proof system and the existence of an optimal acceptor does not work in the heuristic framework. The following questions seem to be natural:

1. Are there interesting examples of heuristic proof systems that have short proofs for more statements than any classical proof system has?

2. Is it possible to construct an optimal heuristic proof system? Possibly for a restricted class of distributional problems.

3. The paper [HIMS12] shows that if one-way functions exist then there is a distributional problem (L, D) with a polynomial-time samplable distribution D concentrated on the complement of L such that (L, D) does not have a polynomially bounded randomized heuristic acceptor. Is there any reasonable assumption that implies the existence of a hard problem for distributional proof system?

In this paper we address the first question in the list above and give examples of heuristic proof systems and nontrivial examples of problems in HeurNP, the second and third questions are left open.

Results. We consider the language $X = \cup_{n\in\mathbb{N}}\{(C, r) \mid C$ is a circuit with n inputs, $r \in \{0,1\}^n, \sharp C > r\}$, where $\sharp C$ is the number of satisfying assignments of the circuit C. The language X is PP-complete, thus $X \notin$ P unless P = NP. For every ensemble of distributions $D = \{D_n\}_{n=1}^{\infty}$ concentrated on the circuits of polynomial in n size and with n inputs we prove that the distributional problem $(X, D \times U)$ is in the class HeurP.

Similarly, we consider the language $Y = \cup_{n\in\mathbb{N}}\{(C, r) \mid C$ is a non-deterministic circuit with n inputs, $r \in \{0,1\}^n, \sharp C > r\}$, where $\sharp C$ is a number of satisfying assignments of the non-deterministic circuit C. This language is PP $\cdot$ NP-complete. Since the class PP $\cdot$ NP is closed under the complement, the assumption NP $\neq$ coNP implies that the language Y is not in NP. For every ensemble of distributions $D = \{D_n\}_{n=1}^{\infty}$ concentrated on non-deterministic circuits of polynomial in n size and with n inputs we prove that the distributional problem $(Y, D \times U)$ is in the class HeurNP. Moreover, we show that if there is a distributional problem in the class (NP, PSamp) that is not in HeurBPP (it is a standard assumption in the average-case complexity theory), then there is a polynomial-time samplable ensemble of distributions D'' such that the distributional problem $(Y, D'' \times U)$ is not in HeurBPP but still $(Y, D'' \times U) \in$ HeurNP.

As mentioned above the problem $(Y, D'' \times U)$ has a proof system that is polynomially bounded for almost all elements of Y. However, this example has the following disadvantages: 1) It is unlikely that Y is in coNP, while almost all classical proof systems are designed for coNP languages. Although Y is coNP-hard and every language $L \in$ coNP can be reduced to Y. But the measure of the image of this reduction may be arbitrary small and all these items may have large proofs. 2) Although the distribution $D'' \times U$ is polynomial-time samplable, it is unclear if it is possible to construct a polynomial-time samplable distribution concentrated on the complement of Y, such distributional problems were considered in [HIMS12]. Our goal is to get an example without these disadvantages.

For every language $L \in$ AM we show that there is a polynomial q such that for every ensemble of distributions D_n concentrated on the strings of a polynomial in n length the distributional problem $(pad_q(L), D \times U^q)$ belongs to the class HeurNP, where $pad_q(L) = \{(x, r) \mid x \in L, r \in \{0,1\}^*, |r| \geq q(|x|)\}$, and $D \times U^q$ is a distribution on pairs in which the first element is distributed according to D, and the second element is distributed uniformly. It is clear that the language $pad_q(L)$ is in NP if and only if the language L is in NP. We also show that if $(pad_q(L), D \times U^q) \in$ HeurBPP then $(L, D) \in$ HeurBPP. We show that there is a heuristic proof system for a distributional proving problem $(pad_q(L), D \times U^q)$, where D is an arbitrary polynomially bounded ensemble of distributions concentrated on the complement of L.

The language GNI that consists of pairs of non-isomorphic graphs with the same number of vertices is a representative of AM. It is clear that GNI $\in$ coNP since the certificate of $(G_1, G_2) \notin$ GNI is a permutation that represents the isomorphism between G_1 and G_2. The question GNI $\stackrel{?}{\in}$ NP is still open so we don't know how to certify efficiently the fact that two graphs are not isomorphic. The result of Klivans and Melkebeek [KvM99]

states that if the complexity class EXP does not coincide with the third level of the polynomial hierarchy Σ_3^{P}, then for infinitely many lengths $\mathtt{GNI}$ has proofs of membership of subexponential size.

We consider the language $pad_q(\mathtt{GNI})$; $pad_q(\mathtt{GNI}) \in$ coNP since $\mathtt{GNI} \in$ coNP. Our results imply that there is a polynomially bounded heuristic proof system for $pad_q(\mathtt{GNI})$ with an arbitrary distribution on pairs of isomorphic graphs and the uniform distribution on strings r. We show that $(pad_q(\mathtt{GNI}), D \times U)$ has a polynomially bounded randomized acceptor if and only if $(\mathtt{GNI}, D)$ has one. Based on the examples of hard instances for the graph isomorphism problem from [vDM10] we give an example of such a distribution D concentrated on pairs of isomorphic graphs such that it is unclear (as far as we know, no one can do it) how to construct a polynomially bounded randomized acceptor for $(\mathtt{GNI}, D)$.

1.4 Technique

In the proof of these results we use Boolean samplers from [Gol11].

A Boolean sampler is a randomized algorithm S, which takes on the input an integer number n and rational numbers ϵ, δ. S has an oracle access to a function $f : \{0,1\}^n \to \{0,1\}$; S makes several non-adaptive requests to the function f and outputs a number in the range $[0,1]$. Let's denote $\overline{f} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)$. For every function $f : \{0,1\}^n \to \{0,1\}$ the following inequality should be satisfied: $\Pr[|S^f(n,\epsilon,\delta) - \overline{f}| \geq \epsilon] < \delta$. A Boolean sampler is called averaging if it outputs the average value of requested values.

Theorem 1.1 ([Gol11])**.** There is an averaging Boolean sampler S which uses n random bits, makes $O(\frac{1}{\epsilon^2 \delta})$ requests to the function, and runs in polynomial in $n, \frac{1}{\epsilon}, \frac{1}{\delta}$ time.

The construction from the Theorem 1.1 is based on the Ramanujan's graph with 2^n vertices. The sampler chooses a random vertex of the graph and returns the average value of the function on the neighbors of the chosen vertex.

A sampler is called monotone if changing the value of the function f in one point from 0 to 1 can not decrease the approximation returned by the sampler. Note that all averaging samplers are monotone.

Theorem 1.2 ([Gol11])**.** There is a monotone Boolean sampler S that uses $n + O(\log \frac{1}{\delta})$ random bits, makes $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ requests to the function and its running time is a polynomial in $n, \frac{1}{\epsilon}, \log \frac{1}{\delta}$. Moreover, if the identically zero function is given to the sampler as an input, then the sampler returns 0 with probability 1.

The construction of the sampler from Theorem 1.2 is the following: we execute several times a sampler from Theorem 1.1 with fixed $\delta = \delta_0$, where δ_0 is an absolute constant and return the median of received answers. Random bits for several executions are used in an economical way by means of a random walk on an expander [IZ89].

In the proof of the membership to the classes HeurP and HeurNP we use a method similar to one used by Pervyshev [Per07] in the proof of the time hierarchy theorem for heuristic classes.

Paper organization. In Section 2 we give basic definitions, in Section 3 we give non-trivial examples of languages in HeurP and HeurNP. In Section 4 we show that every language from the class AM with a padding and every distribution that is uniform over padding is in HeurNP, and we construct a polynomially bounded heuristic proof system for languages from AM with a padding. In Section 5 we generalize this results for other computational models.

2 Preliminaries

An ensemble of distributions $\{D_n\}_{n=1}^{\infty}$ is a sequence of distributions $D_i : \{0,1\}^* \to [0,1]$ with a finite support. Here n is a complexity parameter, it is usually called a security parameter in cryptography. A support of a distribution D_n is the set of strings such that $D_n(x) > 0$, we denote it as $\operatorname{supp} D_n$.

The ensemble U denotes the uniform ensemble, i.e. $U_n(x) = 2^{-n}$ for all $x \in \{0,1\}^n$.

For two ensembles of distributions D and F we define an ensemble $\alpha D + (1-\alpha)F$ for $0 \le \alpha \le 1$ such that $(\alpha D + (1-\alpha)F)_n(x) = \alpha D_n(x) + (1-\alpha)F_n(x)$. We also define an ensemble of distributions $D \times F$ concentrated on pairs of strings as follows: $(D \times F)_n(x,y) = D_n(x)F_n(y)$. In particular $D \times U$ is an ensemble such that $(D \times U)_n(x,r) = D_n(x)2^{-n}$ if $|r| = n$. If $g : \{0,1\}^* \to \{0,1\}^*$ and D is an ensemble of distributions, then $g(D)$ denotes an ensemble of distributions, where $g(D)_n(x) = \sum\limits_{y \in g^{-1}(x)} D_n(y)$. Similarly if $h : \{0,1\}^* \times \mathbb{N} \to \{0,1\}$, then $h(D)$ is an ensemble of distributions, where $h(D)_n(x) = \sum\limits_{y:h(y,n)=x} D_n(y)$.

An ensemble is polynomially bounded if there exists a polynomial p such that for every $x \in \operatorname{supp} D_n$, $|x| \le p(n)$. An ensemble D_n is called polynomial-time samplable if there exists a polynomial-time randomized algorithm G such that $G(1^n)$ is distributed according to D_n. The set of polynomial-time samplable ensembles we denote as PSamp. Every polynomial-time samplable ensemble is polynomially bounded.

A distributional (decision) problem is a pair (L, D) where L is a language and D is an ensemble of distributions.

In this paper we consider three basic computational models: deterministic Turing machines, non-deterministic Turing machines, and randomized Turing machines with bounded error. We assume that these computational models output only 0 or 1. The notions of running time are defined for each of these models: for a deterministic machine it is simply its running time, for a non-deterministic machine it is the maximum running time for all non-deterministic choices, and for a randomized machine with bounded error it is the maximum for all randomized bits. We define a response on a given input for each of these computational models. For a randomized machine with bounded error a response on a given input is a response, which is accepted with the probability at least $\frac{3}{4}$ (such response may not exist, in this case we assume that the response of this machine is $\perp$). The classes $\mathrm{P}, \mathrm{NP}, \mathrm{BPP}$ can be defined as sets of languages, which are decided in polynomial time by deterministic, non-deterministic, randomized with bounded error Turing machines respectively.

Definition 2.1. HeurP (HeurNP, HeurBPP) is a class of distributional decision problems (L, D) that have a deterministic (non-deterministic and randomized, respectively)

algorithm $A(x, \delta; n)$ and a polynomial p such that:

1. For all $x \in \operatorname{supp} D_n$ the running time of $A(x, \delta; n)$ is bounded by $p(\frac{n}{\delta})$;
2. $\Pr_{x \leftarrow D_n}[A(x, \delta; n) \neq L(x)] < \delta$, in the case of a randomized algorithm the probability is taken also over random bits of the algorithm A.

The following definition is a deterministic version of a randomized heuristic proof system from [HIMS12].

Definition 2.2. A heuristic proof system for a distributional problem (L, D) is an algorithm $\Pi(x, w, \delta; n)$ that satisfies the following properties:

1. For every $x \in L$ there is $w \in \{0,1\}^*$, such that $\Pi(x, w, \delta; n) = 1$;
2. $\Pr_{x \leftarrow D_n}[x \notin L \wedge \exists w \Pi(x, w, \delta; n) = 1] < \delta$;
3. The running time of $\Pi(x, w, \delta; n)$ is bounded by a polynomial in $\frac{n+|w|}{\delta}$.

A heuristic proof system is called polynomially bounded if there exists such a polynomial p that for every $x \in L$ there is w that $\Pi(x, w, \delta; n) = 1$ and the size of w is bounded by $p(\frac{n}{\delta})$.

[HIMS12] also defines a notion of a randomized heuristic acceptor.

Definition 2.3. A randomized heuristic acceptor for distributional problem (L, D) is an algorithm $A(x, \delta; n)$ that satisfies the following properties:

1. For every $x \in L$ $A(x, \delta; n) = 1$;
2. $\Pr_{x \leftarrow D_n, A}[x \notin L \wedge A(x, \delta; n) = 1] < \delta$.

An acceptor is called polynomially bounded if for all $x \in L$ the median running time of $A(x, \delta; n)$ is bounded by some polynomial in $\frac{n}{\delta}$, where the median running time of a randomized algorithm is the minimal number t such that the algorithm runs in at most t steps with probability at least $\frac{1}{2}$.

A distributional proving problem [HIMS12] is a distributional problem (L, D) where D is an ensemble of distributions concentrated on the complement of the language L.

3 Examples of problems in HeurP and HeurNP

Let $f : \{0,1\}^* \to \{0,1\}$ be a function. Denote the language $L_f = \cup_{n \in \mathbb{N}} \{r \in \{0,1\}^n \mid 0.r < \overline{f_n}\}$, where $\overline{f_n} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)$.

Theorem 3.1. (1) If f is computable in polynomial time then $(L_f, U) \in$ HeurP. (2) If f is computable in polynomial time by a non-deterministic algorithm (i.e. the language $\{x \in \{0,1\}^* \mid f(x) = 1\} \in$ NP) then $(L_f, U) \in$ HeurNP.

Proof. We describe the algorithm $A(x, \delta)$ (we can omit n as it equals $|x|$). The construction and analysis are similar to items (1) and (2) of the theorem. Let S be an averaging Boolean sampler from Theorem 1.1.

(1) If f is computable in polynomial time, then we define $A(x, \delta)$ as follows:

- If $\delta \leq 2^{-n+2}$ then compute $\overline{f}$, and output 1 if $0.x < \overline{f_n}$, and 0 otherwise.
- If $\delta > 2^{-n+2}$ then run the sampler $S^f(n, \delta/4, \delta/4)$ that uses the string x instead of random bits. Let ν_n be its result. If $\nu_n > 0.x$ then return 1 else return 0.

(2) If f is computable in polynomial time by a non-deterministic algorithm, then we define $A(x, \delta)$ as follows:

- If $\delta \leq 2^{-n+2}$ then guess witnesses for all $x \in \{0,1\}^n$, compute $\overline{f}$ and return 1, if $0.x \leq \overline{f_n}$, and 0 otherwise.
- If $\delta > 2^{-n+2}$ then a witness of the algorithm consists of q strings $w_1, w_2, \ldots, w_q$, where $q = O(\frac{1}{\delta^3})$ is a number of requests made by the sampler $S^f(n, \delta/4, \delta/4)$ using the string x as random bits. Execute the sampler $S^f(n, \delta/4, \delta/4)$ that uses the string x as random bits, the witness w_i is used to compute the function f in the i-th point. Let ν_n be the sampler's output. If $\nu_n > 0.x$ then return 1 else return 0. Notice that the sampler is monotone as it averages requested values. Thus if some witnesses are wrong and some values become zeros instead of ones, then the result of ν_n can only decrease, i.e. the answer can change from 1 to 0, but not vice versa.

If $\delta \leq 2^{-n+2}$, then the algorithm outputs the right answer (in the case of non-deterministic algorithm we assume that witnesses for all x such that $f(x) = 1$ are correct) and the running time of the algorithm equals $2^n poly(n)$, and hence to a polynomial in $\frac{n}{\delta}$.

In the second case the running time of $A(x, \delta)$ is bounded by a polynomial in n and $\frac{1}{\delta}$, since the function f is polynomial-time computable (in item (2) by non-deterministic algorithm) and S is a sampler. We estimate the fraction of inputs x such that the algorithm $A(x, \delta)$ outputs the wrong answer. The set of such inputs is contained in the union of the following two sets:

1. The set of $x \in \{0,1\}^n$ such that the sampler S does not output a $\delta/4$ approximation to $\overline{f_n}$. The uniform measure of this set does not exceed $\frac{\delta}{4}$ by the definition of the sampler.
2. The set of $x \in \{0,1\}^n$ such that $|0.x - \overline{f_n}| \leq \delta/4$. The uniform measure of this set does not exceed $\frac{\delta}{2} + \frac{1}{2^n}$.

Altogether the uniform measure of the set of $x \in \{0,1\}^n$ such that $A(x, \delta) \neq L_f(x)$ does not exceed $\frac{3}{4}\delta + 2^{-n} < \delta$ for $\delta > 2^{-n+2}$. □

Consider the language $X = \cup_{n \in \mathbb{N}} \{(C, r) \mid C \text{ is a circuit with } n \text{ inputs}, r \in \{0,1\}^n, \sharp C > r\}$, where $\sharp C$ is a number of satisfying assignments of the circuit C.

Recall that the complexity class PP consists of languages L that have a randomized polynomial-time Turing machine M such that for every x, $x \in L \iff \Pr_M[M(x) = L(x)] \geq 1/2$.

Proposition 3.1. The language X is PP-complete.

Proof. We first show that $X \in \mathrm{PP}$. Let's describe a randomized PP-algorithm. Let the input be a circuit $C(x_1, x_2, \ldots, x_n)$ and a number $0 \le r \le 2^n - 1$. Consider the following predicate $P(b, x_1, x_2, \ldots, x_n) = (b \wedge C(x_1, x_2, \ldots, x_n)) \vee ((\overline{x_1 x_2 \ldots x_n} \le 2^n - r - 1) \wedge \neg b)$, where $\overline{x_1 x_2 \ldots x_n}$ is a number in the binary representation.

- The predicate P depends on $n + 1$ Boolean variables.
- The predicate P is computable in polynomial time.
- The number of satisfying assignments of the predicate P equals $\sharp C + 2^n - r - 1$. Thus, $\sharp C > r$ if and only if the number of satisfying assignments of P is at least 2^n.

The PP-algorithm is the following: take a random assignment to $b, x_1, x_2, \ldots, x_n$ and return $P(b, x_1, x_2, \ldots, x_n)$.

Now we show that any language of PP is reducible to X. Let $L \in \mathrm{PP}$. Let M be a polynomial-time randomized Turing machine such that for every x, $x \in L$ if and only if $\Pr[M(x) = 1] \ge \frac{1}{2}$. The machine M on the input x uses $q(|x|)$ random bits where q is a polynomial. We construct a circuit C that has $q(|x|)$ inputs and simulates the machine M on the input x using random bits from the input. Finally, the reduction maps $x \mapsto (C, 2^{q(|x|)-1} - 1)$. □

A non-deterministic circuit is a circuit whose input vertices are divided into two parts: the vertices from the first part we call inputs, the vertices from the second part we call witnesses. The value of such circuit on the input x equals 1, if there is a witness y such that $C(x; y) = 1$. We also define $C(x) = \exists y C(x; y)$.

Consider the language $Y = \cup_{n \in \mathbb{N}} \{(C, r) \mid C \text{ is a non-deterministic circuit with } n \text{ inputs}, r \in \{0,1\}^n, \sharp C > r\}$, where $\sharp C$ is a number of satisfying assignments of the circuit C, in other words, if the circuit C uses m bits of the witness, then $\sharp C = |\{x \in \{0,1\}^n \mid \exists y \in \{0,1\}^m : C(x, y) = 1\}|$.

Definition 3.1. The class $\mathrm{C} \cdot \mathrm{NP}$ [Wag86, Tod91] consists of languages L that have a randomized polynomial-time Turing machine M such that for every x, $x \in L$ if and only if $\Pr[M(x) \in \mathtt{SAT}] \ge \frac{1}{2}$, where $\mathtt{SAT}$ is the language of satisfiable propositional formulas.

Proposition 3.2. The language Y is $\mathrm{C} \cdot \mathrm{NP}$-complete.

Proof. The proof is similar to the proof of Proposition 3.1. Let's describe the C-reduction of the language Y to $\mathtt{SAT}$. Let the input be a non-deterministic circuit C with n inputs and a number $0 \le r \le 2^n - 1$. We construct the predicate $P(b, x_1, x_2, \ldots, x_n) = \exists w (b \wedge C(x_1, x_2, \ldots, x_n; w)) \vee ((\overline{x_1 x_2 \ldots x_n} \le 2^n - r - 1) \wedge \neg b)$.

If we fix values of the variables $b, x_1, x_2, \ldots, x_n$ then it is easy to construct a formula $\phi_{b,x}$ that is satisfiable if and only if $P(b, x_1, x_2, \ldots, x_n) = 1$. The randomized machine picks random values of $b, x_1, x_2, \ldots, x_n$ and returns the corresponding formula $\phi_{b,x}$.

Now we show that every language $L \in \mathrm{C} \cdot \mathrm{NP}$ can be reduced to Y. Let M be a randomized C-reduction from L to $\mathtt{SAT}$. Let a circuit C on an input r compute a propositional formula $C(r)$. And let a circuit A given strings y, r and a description of a circuit C as an input returns the result of the substitution of the values from the string

y into the formula $C(r)$. Let C_x be a circuit that simulates a machine M on an input x, inputs of this circuit are random bits of the machine M. The circuit C on the input r computes a propositional formula $C_x(r)$. Let a non-deterministic circuit A_{C_x} take a string r as an input and return 1 if and only if there is such a string y that the result of substitution of this string into the formula $C_x(r)$ equals 1. The reduction from L to Y maps x to $(A_{C_x}(r; y), 2^{q(n)-1} - 1)$. □

Theorem 3.2. (1) Let D be a polynomially bounded ensemble and let the support of D_n consist of circuits with n inputs, then $(X, D \times U) \in \mathrm{HeurP}$. (2) Let D be a polynomially bounded ensemble and let a support of D_n consist of non-deterministic circuits with n inputs, then $(Y, D \times U) \in \mathrm{HeurNP}$.

Proof. We prove the statement of item (1), the proof of item (2) is similar. We describe an algorithm $B(x, \delta; n)$. We consider the set of strings with nonzero probability according to the distribution $D \times U$. The string x has the form (C, r), where C is a circuit with n inputs and $r \in \{0,1\}^n$; the size of C is bounded by the polynomial in n, since the ensemble D is polynomially bounded. Let $A_C(x, \delta)$ be the algorithm from the proof of Theorem 3.1 that decides the language L_{g_C}, where g_C is computed by the circuit C on inputs of length n and is zero on inputs of other lengths. By construction the algorithm A_C uses the circuit C as an oracle, therefore its running time is a polynomial in the size of the circuit C and $\frac{n}{\delta}$.

The algorithm $B((C, r), \delta; n)$ executes $A_C(r, \delta)$ and returns its answer. It is clear that the running time of B is bounded by the polynomial in $\frac{n}{\delta}$.

$\Pr_{(C,r)\leftarrow(D\times U)_n}[B((C, r), \delta; n) \neq X(C, r)] = \mathrm{E}_{C\leftarrow D_n}[\Pr_{r\leftarrow U_n}[A_C(r, \delta) \neq L_{g_C}(r)]] < \delta$.

□

Proposition 3.3. For every problem $(L, D) \in (\mathrm{NP}, \mathrm{PSamp})$ one can construct such a distribution $D'' \in \mathrm{PSamp}$ that if $(Y, D'' \times U) \in \mathrm{HeurBPP}$, then $(L, D) \in \mathrm{HeurBPP}$.

Proof. Consider an arbitrary distributional problem $(L, D) \in (\mathrm{NP}, \mathrm{PSamp})$. Consider a reduction g from L to `CircuitSAT`, where `CircuitSAT` is the language of satisfiable circuits. Let the distribution $D' = g(D)$ correspond to the distribution of images of the function g with inputs distributed according to D. It is clear that if $(\texttt{CircuitSAT}, D') \in \mathrm{HeurBPP}$, then $(L, D) \in \mathrm{HeurBPP}$. Let a function $h : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$ map circuits to non-deterministic circuits in such a way that the old input becomes a witness and the new circuit contains n dummy inputs, that is the circuit does not depend on such inputs. This means that the function $h(C, n) = E$, where $E(y; x) = C(x)$ and the size of y equals n. The circuit depends on the witness in the same way as the old circuit. Let $D'' = h(D')$. The support of the distribution D''_n consists of non-deterministic circuits $h(C)$ with n inputs, where C is a deterministic circuit. If the circuit C is satisfiable, then for all $0 \leq r \leq 2^n - 1$, $(h(C), r) \in Y$ and, on the contrary, if C is unsatisfiable, then for all $0 \leq r \leq 2^n - 1$, $(h(C), r) \notin Y$.

Let an algorithm $A(x, \delta; n)$ solve the problem $(Y, D'' \times U)$ of the class HeurBPP, let's describe an algorithm $B(x, \delta; n)$ that generates a random string $r \in \{0,1\}^n$ and executes the algorithm $A((h(g(x)), r), \delta; n)$.

$\Pr_{x\leftarrow D_n}[B(x, \delta; n) \neq L(x)] = \Pr_{x\leftarrow D_n, r\leftarrow U_n}[A((h(g(x)), r), \delta; n) \neq Y(x, r)] = \Pr_{x\leftarrow D''_n, r\leftarrow U_n}[A(x, \delta; n) \neq Y(x, r)] < \delta$. □

Corollary 3.1. There is such a polynomial-time samplable distribution D'' that the inclusion $(Y, D'' \times U) \in \mathrm{HeurBPP}$ implies that $(\mathrm{NP}, \mathrm{PSamp}) \subseteq \mathrm{HeurBPP}$.

Proof. We apply Proposition 3.3 to a complete problem in $(\mathrm{NP}, \mathrm{PSamp})$. A complete problem in $(\mathrm{NP}, \mathrm{PSamp})$ under the reductions under which HeurBPP is closed, is constructed, for example, in [IL90] (see also [BT06]). □

Theorem 3.3. The distributional problem $(Y, D'' \times U)$ has the following properties:

1. $(Y, D'' \times U) \in \mathrm{HeurNP}$;
2. if $\mathrm{NP} \neq \mathrm{coNP}$, then $Y \notin \mathrm{NP}$;
3. if $(\mathrm{NP}, \mathrm{PSamp}) \not\subseteq \mathrm{HeurBPP}$, then $(Y, D'' \times U) \notin \mathrm{HeurBPP}$.

Proof.

1. D'' is polynomially bounded since it is polynomial-time samplable, therefore Theorem 3.2 implies that $(Y, D'' \times U) \in \mathrm{HeurNP}$.
2. Assume that $\mathrm{NP} \neq \mathrm{coNP}$. By Proposition 3.2 the language Y is $\mathrm{C} \cdot \mathrm{NP}$ complete, therefore if $Y \in \mathrm{NP}$, then $\mathrm{PP} \subseteq \mathrm{C} \cdot \mathrm{NP} \subseteq \mathrm{NP}$, therefore $\mathrm{NP} = \mathrm{PP}$. Class PP is closed under the complement, hence $\mathrm{NP} = \mathrm{coNP}$ and this is a contradiction.
3. Follows from Corollary 3.1.

□

4 Heuristic proof system for AM

We recall the definition of the class AM. A language L is in the class AM if there is a polynomial-time non-deterministic machine M and a polynomial q such that for every string x the following holds:

- if $x \in L$, then $\Pr_{z \leftarrow U_{q(|x|)}}[M(x, z) = 1] > \frac{3}{4}$,
- if $x \notin L$, then $\Pr_{z \leftarrow U_{q(|x|)}}[M(x, z) = 1] < \frac{1}{4}$.

It is known that for $x \in L$ the error can be replaced by zero, i.e. if the language L is in AM then there is a polynomial-time non-deterministic machine M and a polynomial q such that for every string x the following holds:

- if $x \in L$, then $\Pr_{z \leftarrow U_{q(|x|)}}[M(x, z) = 1] = 1$,
- if $x \notin L$, then $\Pr_{z \leftarrow U_{q(|x|)}}[M(x, z) = 1] < \frac{1}{4}$.

We can also consider languages from the class AM as protocols between Arthur and Merlin. Merlin tries to convince Arthur that $x \in L$. The protocol of their interaction is as follows: first Arthur sends a random string of length $q(|x|)$ to Merlin, then Merlin sends a witness y for a non-deterministic machine M on the input (x, z) to Arthur, Arthur runs the machine $M(x, y)$ with the witness y and returns its answer. If $x \in L$ then there

exists Merlin that convinces Arthur to accept with the probability more than $\frac{3}{4}$, and if $x \notin L$ then every Merlin convinces Arthur with the probability less than $\frac{1}{4}$.

We show that these requirements can be made stronger and the error probability can be equal to $2^{-p(|x|)}$, with only $q(|x|) + O(p(|x|))$ random bits used by Arthur.

Lemma 4.1. If L is in the class AM then there is a polynomial q and a constant $c > 0$ such that for every polynomial p there is a polynomial-time non-deterministic machine M, such that for every string x the following conditions are satisfied:

- if $x \in L$, then $\Pr_{z \leftarrow U_{q(|x|)+cp(|x|)}}[M(x, z) = 1] = 1$,
- if $x \notin L$, then $\Pr_{z \leftarrow U_{q(|x|)+cp(|x|)}}[M(x, z) = 1] < 2^{-p(|x|)}$.

Proof. Consider a machine M' and a polynomial q such that for every string x the following holds:

- if $x \in L$, then $\Pr_{z \leftarrow U_{q(|x|)}}[M'(x, z) = 1] = 1$,
- if $x \notin L$, then $\Pr_{z \leftarrow U_{q(|x|)}}[M'(x, z) = 1] < \frac{1}{4}$.

We consider the sampler $S(n, \epsilon, \delta)$ from Theorem 1.2, let c be such a constant that S uses $n + c \log \frac{1}{\delta}$ random bits. Let's describe how the non-deterministic machine M operates on the input (x, r), where $|z| = q(|x|) + cp(|x|)$. M executes the sampler S with the parameters $\delta = 2^{-p(|x|)}$, $\epsilon = \frac{1}{8}$ to compute the average value of $M'(x, r)$ over all strings r of size $|z| = q(|x|)$ using the string z as random bits. Every time when S requests the value of $M'(x, z)$, the witness for M' is guessed. We denote the return value as ν. If $\nu < \frac{1}{2}$ then M returns 0 else it returns 1. If $x \notin L$ then the probability that $\nu \geq \frac{1}{2}$ is less than $\delta = 2^{-p(|x|)}$ since the sampler is monotone. If $x \in L$ then the sampler returns 1 with the probability that is equal to 1 for correctly guessed witnesses for all requests of values of $M'(x, z)$. □

Let q be a polynomial, for every language L we denote $pad_q(L) = \{(x, r) \mid x \in L, r \in \{0, 1\}^*, |r| \geq q(|x|)\}$.

Theorem 4.1. Let a language $L \in$ AM and let an ensemble D be polynomially bounded, then there is a polynomial g such that $(pad_g(L), D \times U^{(g)}) \in$ HeurNP, where $U_n^{(g)} = U_{g(n)}$.

Proof. Let M be the non-deterministic Turing machine from Lemma 4.1 constructed for the language L and a polynomial p that is equal to the polynomial q, and assume $g(n) = (c + 1)q(n)$.

Without loss of generality we assume that the polynomial q is an upper bound for the number of random bits that the Arthur-Merlin protocol needs for all $x \in \operatorname{supp} D_n$.

We describe a non-deterministic algorithm $A(x, \delta; n)$ that solves the problem $(pad_g(L), D \times U^{(g)})$. We assume that x has the form (y, r), where $|r| = (c + 1)q(n)$ (the probability of any other string is zero). If $\delta < 2^{-q(n)}$ then the algorithm goes over all $z \in \{0, 1\}^{q(n)(c+1)}$, guesses a witness for the machine M on all inputs (y, z). If all the witnesses are guessed correctly, the algorithm accepts, else rejects.

If $\delta \geq 2^{-q(n)}$, then the algorithm executes the machine M on the input (y, r) and returns the answer. For every x the fraction of strings r on which the answer is wrong is less than $2^{-q(n)} = \delta$.

If $\delta < 2^{-q(n)}$, then the non-deterministic algorithm $A((y, r), \delta; n)$ accepts if $y \in L$, and rejects otherwise. The running time is bounded by $(\frac{n}{\delta})^{c+1}$. If $\delta \geq 2^{-q(n)}$, then the running time is bounded by the running time of the machine M on the support of D_n i.e. by a polynomial in n. □

Corollary 4.1. Let a language L be in AM, let D be a polynomially bounded ensemble concentrated on the complement of L, then there is a polynomial q such that for the distributional proving problem $(pad_q(L), D \times U^{(q)})$ there is a polynomially bounded proof system.

Proof. The algorithm $A((y, r), \delta; n)$ constructed in the proof of Theorem 4.1 is precisely the proof system we need. A proof in this proof system is a non-deterministic witness. It is clear from the proof that for every element of the language there is a witness of size $poly(\frac{n}{\delta})$, and the condition on the error on the complement of the language follows directly from Theorem 4.1. □

Remark 4.1. It may seem strange that the size of the padding is not just a polynomial in the size of x. We consider distributions of the form $D \times U^{(g)}$, so the size of padding of inputs that have non-zero probability is equal to a polynomial in complexity parameter n that is not necessary equal to $|x|$.

Note that for every polynomial q the language L is polynomial-time many-to-one reducible to the language $pad_q(L)$ and vice versa. Therefore $pad_q(L) \in$ NP if and only if $L \in$ NP.

Indeed, one can prove the existence of an average-case reduction:

Proposition 4.1. For every polynomially bounded ensemble D and every polynomial q, if $(pad_q(L), D \times U^{(q)}) \in$ HeurBPP, then $(L, D) \in$ HeurBPP.

Proof. Let an algorithm $B(x, \delta; n)$ solve the problem $(pad_q(L), D \times U^{(q)})$ in the class HeurBPP. We describe an algorithm $A(x, \delta; n)$ that solves the problem (L, D). It picks a random string r of length $q(n)$ and executes the algorithm $B((x, r), \delta; n)$. □

We consider the language $\texttt{GNI} \in$ AM that consists of pairs of nonisomorphic graphs with the same number of vertices. $\texttt{GNI} \in$ coNP, therefore for any polynomial q, $pad_q(\texttt{GNI}) \in$ coNP. By the Corollary 4.1, for any polynomially bounded distribution D concentrated on the complement of GNI, there exists a polynomial q such that distributional proving problem $(pad_q(\texttt{GNI}), D \times U^q)$ has polynomially bounded heuristic proof system. We will show that there exists such a polynomial-time samplable D that it is not obvious that $(pad_q(\texttt{GNI}), D \times U^q)$ has a polynomially bounded randomized heuristic acceptor.

Similarly to Proposition 4.1 one can show that if $(pad_q(\texttt{GNI}), D \times U^q)$ has a polynomially bounded randomized heuristic acceptor, then $(\texttt{GNI}, D)$ also has one.

Lemma 4.2. Let D be an ensemble of distributions such that D_n is concentrated on pairs of isomorphic graphs with n vertices. If $(\texttt{GNI}, D)$ has a polynomially bounded randomized heuristic acceptor, then there exists a randomized polynomial-time algorithm B such that for every $x \in \texttt{GNI}$, $\Pr[B(x) = 1] \geq \frac{3}{4}$ and $\Pr_{B, x \leftarrow D_n}[B(x) = 1] < \frac{1}{n^{100}}$.

Proof. Let $A(x, \delta)$ be a polynomially bounded randomized heuristic acceptor for $(\mathtt{GNI}, D)$ and its median running time on the elements of $\mathtt{GNI}$ is bounded by a polynomial $p(n/\delta)$. Let an algorithm $A'(x, \delta)$ execute two copies in parallel of $A(x, \delta/2)$ and return 1 if one of this copies stops and returns 1. It is straightforward that A' is also a randomized heuristic acceptor for $(\mathtt{GNI}, D)$ and there is a polynomial p' such that $A'(x, \delta)$ runs in at most $p'(n/\delta)$ steps with probability $\frac{3}{4}$. Let B execute $A'(x, n^{-100})$ in $p'(n^{101})$ steps and return 0 if $A'(x, n^{100})$ did not stop in $p'(n^{101})$ steps. □

There is an example of a distribution D for which nowadays nobody knows how to construct an algorithm B with properties from Lemma 4.2. The paper [vDM10] gives an example of a distribution on graphs such that the problem of isomorphism is hard for two graphs from this distribution. The distribution H_q is based on an affine plane of order q. Graphs from H_q have q^2 vertices that correspond to points of an affine plane. Every affine plane of order q has $q + 1$ classes of parallel lines: $C_1, C_2, \ldots, C_{q+1}$. A random graph G from H_q is defined by a random subset I of $\{1, 2, \ldots, q + 1\}$ such that $|I| = \lceil \frac{q+1}{2} \rceil$. Two vertices a and b from G are connected by an edge if the line (a, b) belongs to C_i for some $i \in I$. A distribution D may be constructed from H in the following way: pick a random graph G from H_q and two random permutations π_0, π_1, then output $(\pi_0(G), \pi_1(G))$. If there exists an algorithm B from the statement of Lemma 4.2, then B gives the correct answer (with probability $\frac{3}{4}$) for all nonisomorphic pairs of graphs from H_q and works almost correct for isomorphic pairs from D. As far as we know, nowadays there are no known algorithms with such properties.

5 Some generalizations

Edward A. Hirsch noted that Theorem 4.1 may be generalized as follows.

Let $\mathfrak{C}$ be some computational model; for every input a $\mathfrak{C}$-machine either accepts or rejects; we assume that the notion of a $\mathfrak{C}$-machine running time on a given input is well defined. We denote the set of languages that may be decided by $\mathfrak{C}$-machines in $O(t(n))$ steps by $\mathfrak{C}\mathrm{Time}[\mathrm{t(n)}]$. We also claim that the model $\mathfrak{C}$ satisfies the following *property of monotone nonadaptive composition*: let F be a deterministic oracle Turing machine that on every input makes several oracle requests that depend only on the input and not on each other, does some calculations and returns an answer from $\{0, 1\}$. The running time of the machine F on the input x for every oracle answers is at most $f(|x|)$. On every input the answer of the machine F monotonically depends on the oracle answers, that is if we change an answer of the oracle from 1 to 0, the result can not increase. Let on the input x the machine F make oracle requests $y_1, y_2, \ldots, y_{k(x)}$. Let language O be from the class $\mathfrak{C}\mathrm{Time}[\mathrm{h(n)}]$, then the calculations of F with oracle O on the input x may be simulated on a $\mathfrak{C}$-machine in $O(f(n) + h(|y_1|) + \cdots + h(|y_{k(x)}|))$ steps. Note that nondeterministic Turing machines satisfy the property of monotone nonadaptive composition.

Class $\mathfrak{C}\mathrm{P} = \cup_{\mathrm{c}>0}\mathfrak{C}\mathrm{Time}[\mathrm{n^c}]$ is the set of languages that can be decided in polynomial time on $\mathfrak{C}$-machines. Class Heur$\mathfrak{C}$P is defined similarly to Definition 2.1.

For every complexity class A Schöning [Sch89] defines a complexity class BP · A that consists of languages L that have a randomized polynomial-time algorithm B and a language $R \in \mathrm{A}$ that for all $x \in L$, $\Pr[B(x) \in R] \geq \frac{2}{3}$ and for all $x \notin L$, $\Pr[B(x) \notin R] \geq \frac{2}{3}$.

Similarly to Lemma 4.1 we prove the following:

Lemma 5.1. For every $L \in \mathrm{BP} \cdot \mathfrak{C}\mathrm{P}$ there is a polynomial q and a constant $c > 0$ such that for every polynomial p there is a polynomial-time $\mathfrak{C}$-machine M, such that for every string x the following conditions hold:

- if $x \in L$, then $\mathrm{Pr}_{z \leftarrow U_{q(|x|)+cp(|x|)}}[M(x,z) = 1] \geq 1 - 2^{-p(|x|)}$,
- if $x \notin L$, then $\mathrm{Pr}_{z \leftarrow U_{q(|x|)+cp(|x|)}}[M(x,z) = 1] < 2^{-p(|x|)}$.

Proof. Consider a polynomial-time randomized algorithm B and a language $R \in \mathfrak{C}\mathrm{P}$ such that

- if $x \in L$, then $\mathrm{Pr}[B(x) \in R] \geq \frac{2}{3}$,
- if $x \notin L$, then $\mathrm{Pr}[B(x) \in R] < \frac{1}{3}$.

We assume that the algorithm B uses at most $q(|x|)$ random bits for some polynomial q. We consider a language A that consists of pairs (x, r), where $|r| = q(|x|)$, such that $B(x)$ returns an element of R if B uses r as its random bits. Since the model $\mathfrak{C}$ has monotone nonadaptive composition property, $A \in \mathfrak{C}\mathrm{P}$. Let M' be a polynomial-time $\mathfrak{C}$-machine that decides A.

We consider the sampler $S(n, \epsilon, \delta)$ from Theorem 1.2, let c be such a constant that S uses $n + c \log \frac{1}{\delta}$ random bits. We define the language A' that consists of pairs (x, z) where $|z| = q(|x|) + cp(|x|)$. We execute the sampler S with the parameters $\delta = 2^{-p(|x|)}$, $\epsilon = \frac{1}{8}$ to compute the average value of $M'(x, z)$ over all strings z of size $|z| = q(|x|)$ using r as random bits. We denote the return value as ν. If $\nu < \frac{1}{2}$ then assume $(x, z) \in A'$, otherwise assume $(x, z) \notin A'$. Since the sampler is monotone and nonadaptive then by the property of the model $\mathfrak{C}$, A' is in the class $\mathfrak{C}\mathrm{P}$.

If $x \notin L$, then the probability over uniformly distributed z that $\nu \geq \frac{1}{2}$ is less than $\delta = 2^{-p(|x|)}$. If $x \in L$, then the probability over uniformly distributed z that $\nu \leq \frac{1}{2}$ is less than $\delta = 2^{-p(|x|)}$.

Finally, machine M is a polynomial-time $\mathfrak{C}$-machine, that decides A'. □

Similarly to Theorem 4.1 we prove the following:

Theorem 5.1. Let a language $L \in \mathrm{BP} \cdot \mathfrak{C}\mathrm{P}$ and let an ensemble D be polynomially bounded, then there is a polynomial g such that $(pad_g(L), D \times U^{(g)}) \in \mathrm{Heur}\mathfrak{C}\mathrm{P}$, where $U_n^{(g)} = U_{g(n)}$.

Proof. Let M be the $\mathfrak{C}$-machine from Lemma 5.1 constructed for the language L and the polynomial p that is equal to the polynomial q; and assume $g(n) = (c + 1)q(n)$.

Without loss of generality we assume that the polynomial q is an upper bound for the number of random bits that $\mathrm{BP} \cdot \mathfrak{C}\mathrm{P}$ algorithm needs for all $x \in \operatorname{supp} D_n$.

We describe a $\mathfrak{C}$-algorithm $A(x, \delta; n)$ that solves the problem $(pad_g(L), D \times U^{(g)})$. We assume that x has the form (y, r), where $|r| = (c + 1)q(n)$ (the probability of any other string is zero). If $\delta < 2^{-q(n)}$ then the algorithm goes over all $z \in \{0, 1\}^{q(n)(c+1)}$, executes $\mathfrak{C}$-machine M on every string (y, z). If all executions accept, the algorithm accepts, otherwise rejects. The property of monotone nonadaptive composition of the model $\mathfrak{C}$ implies that the algorithm described above is a correct $\mathfrak{C}$-algorithm.

If $\delta \geq 2^{-q(n)}$, then the algorithm executes the machine M on the input (y, r) and returns the answer. For every x the fraction of strings r on which the answer is wrong is less than $2^{-q(n)} = \delta$.

If $\delta < 2^{-q(n)}$, then $\mathfrak{C}$-algorithm $A((y, r), \delta; n)$ accepts if $y \in L$, and rejects otherwise. The running time is bounded by $(\frac{n}{\delta})^{c+1}$. If $\delta \geq 2^{-q(n)}$, then the running time is bounded by the running time of the machine M on the support of D_n, i.e. by a polynomial in n. □

Remark 5.1. It follows from the proof of Theorem 5.1 that the polynomial $g(n)$ depends only on lengths of strings from the support of D_n and that there is one algorithm $A(x, \delta; n)$ that suits for all distributions D corresponding to a given polynomial g. We show that it is unlikely that it is possible to prove the similar result for $\mathrm{C} \cdot \mathfrak{C}\mathrm{P}$, where a complexity class $\mathrm{C} \cdot \mathrm{A}$ [Wag86, Tod91] consists of languages L that have a randomized polynomial-time algorithm B and a language $R \in \mathrm{A}$ such that $x \in L$ iff $\Pr[B(x) \in R] \geq \frac{1}{2}$. Moreover, we show that this is unlikely even for the case when $\mathfrak{C}$ is deterministic Turing machines. Assume that for every language $L \in \mathrm{PP} = \mathrm{C} \cdot \mathrm{P}$ there exists a randomized algorithm $A(x, \delta)$ and a polynomial g such that for all distributions D that have their support D_n concentrated on strings of length n the following holds

1. Running time $A(x, \delta)$ is bounded by a polynomial $p(\frac{|x|}{\delta})$;
2. For every n and δ, $\Pr_{x \leftarrow D_n, r \leftarrow U_{g(n)}, A}[A((x, r), \delta) \neq L(x)] < \delta$.

A distribution D may be concentrated on every particular input x of length n, thus an algorithm $B(x)$ that guesses a random string $r \in \{0,1\}^{g(|x|)}$ and executes $A((x, r), \frac{1}{10})$ will return $L(x)$ with probability $\frac{9}{10}$. Hence $L \in \mathrm{BPP}$ and therefore $\mathrm{PP} \subseteq \mathrm{BPP}$. So we get $\mathrm{NP} \subseteq \mathrm{BPP} \subseteq \mathrm{P/poly}$ and thus by Karp-Lipton theorem the polynomial hierarchy collapses: $\mathrm{PH} = \Sigma_2^{\mathrm{P}}$.

Since $\mathrm{AM} = \mathrm{BP} \cdot \mathrm{NP}$, Theorem 5.1 generalizes Theorem 4.1. Another example of a computational model that can be used in the Theorem 5.1 is $\oplus$-machines. $\oplus$-machine is a nondeterministic Turing machine that accepts input if it has the odd number of accepting computations. The property of monotone nonadaptive composition follows from [PZ83]. The proof of Toda's theorem [Tod91] implies that $\mathrm{PH} \subseteq \mathrm{BP} \cdot \oplus\mathrm{P}$.

Corollary 5.1. Let a language $L \in \mathrm{PH}$ and let an ensemble D be polynomially bounded, then there is a polynomial q such that $(pad_q(L), D \times U^{(q)}) \in \mathrm{Heur}\oplus\mathrm{P}$, where $U_n^{(q)} = U_{q(n)}$.

We also note that Theorem 3.1 can be easily generalized as follows:

Theorem 5.2. If f is computable in polynomial time by a $\mathfrak{C}$-algorithm (i.e. the language $\{x \in \{0,1\}^* \mid f(x) = 1\} \in \mathfrak{C}\mathrm{P}$) then $(L_f, U) \in \mathrm{Heur}\mathfrak{C}\mathrm{P}$.

Acknowledgements

The authors are grateful to Edward A. Hirsch for fruitful discussions and for suggestions to generalize results, to Ilya Ponomarenko for advice concerning hard distributions for graph isomorphism problem, to Olga Khmelevskaya and to an anonymous reviewer for multiple helpful comments that improved the quality of the paper.

References

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundation and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 316–324, 2004.

[Gol11] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 302–332, 2011.

[HIMS12] Edward A. Hirsch, Dmitry Itsykson, Ivan Monakhov, and Alexander Smal. On optimal heuristic randomized semidecision procedures, with applications to proof complexity and cryptography. *Theory of Computing Systems*, 51(2):179–195, 2012. Extended abstract appeared in the proceedings of *STACS-2010*. Preliminary version is available as ECCC TR10-193.

[HINS13] E. A. Hirsch, D. M. Itsykson, V. O. Nikolaenko, and A. V. Smal. Optimal heuristic algorithms for the image of an injective function. *Journal of Mathematical Sciences*, 188(1):7–16, 2013.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Proc. of EUROCRYPT-2005*, 2005.

[IL90] Russell Impagliazzo and Leonid Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1990.

[Its10] Dmitry M. Itsykson. Structural complexity of AvgBPP. *Annals of Pure and Applied Logic*, 162(3):213–223, 2010.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 248–253, Washington, DC, USA, 1989. IEEE Computer Society.

[KP89] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, September 1989.

[KvM99] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, STOC '99, pages 659–667, 1999.

[Mes99] Jochen Messner. On optimal algorithms and optimal proof systems. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 361–372, 1999.

[Per07] Konstantin Pervyshev. On heuristic time hierarchies. In *Proceedings of the IEEE Conference on Computational Complexity*, pages 347–358, 2007.

[PZ83] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings of the 6th GI Conference on Theoretical Computer Science*, volume 145 of *Lecture Notes in Computer Science*, pages 269–276, Berlin, 1983. Springer-Verlag.

[Sch89] Uwe Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39(1):84–100, 1989.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, October 1991.

[vDM10] E.R. van Dama and M. Muzychuk. Some implications on amorphic association schemes. *Journal of Combinatorial Theory, Series A*, 117:111–127, 2010.

[Wag86] Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Inf.*, 23(3):325–356, June 1986.