Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 453-464
www.stacs-conf.org

ON OPTIMAL HEURISTIC RANDOMIZED SEMIDECISION PROCEDURES, WITH APPLICATION TO PROOF COMPLEXITY

EDWARD A. HIRSCH AND DMITRY ITSYKSON

Steklov Institute of Mathematics at St. Petersburg,
27 Fontanka, St.Petersburg, 191023, Russia
URL: http://logic.pdmi.ras.ru/~hirsch
URL: http://logic.pdmi.ras.ru/~dmitrits

Abstract. The existence of a (p-)optimal propositional proof system is a major open question in (proof) complexity; many people conjecture that such systems do not exist. Krajíček and Pudlák [KP89] show that this question is equivalent to the existence of an algorithm that is optimal[1] on all propositional tautologies. Monroe [Mon09] recently gave a conjecture implying that such algorithm does not exist.

We show that in the presence of errors such optimal algorithms *do* exist. The concept is motivated by the notion of heuristic algorithms. Namely, we allow the algorithm to claim a small number of false "theorems" (according to any polynomial-time samplable distribution on non-tautologies) and err with bounded probability on other inputs.

Our result can also be viewed as the existence of an optimal proof system in a class of proof systems obtained by generalizing automatizable proof systems.

1. Introduction

Given a specific problem, does there exist the "fastest" algorithm for it? Does there exist a proof system possessing the "shortest" proofs of the positive solutions to the problem? Although the first result in this direction was obtained by Levin [Lev73] in 1970s, these important questions are still open for most interesting languages, for example, the language of propositional tautologies.

1998 ACM Subject Classification: F.2.

Key words and phrases: propositional proof complexity, optimal algorithm.

Partially supported by grants RFBR 08-01-00640 and 09-01-12066, and the president of Russia grant "Leading Scientific Schools" NSh-4392.2008.1, by Federal Target Programme "Scientific and scientific-pedagogical personnel of the innovative Russia" 2009-2013 (contract N П265 from 23.07.2009). The second author is also supported by Russian Science Support Foundation.

27th Symposium on Theoretical Aspects of Computer Science, Nancy, 2010
Editors: Jean-Yves Marion, Thomas Schwentick
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.STACS.2010.2475

Classical version of the problem. According to Cook and Reckhow [CR79], a proof system is a polynomial-time mapping of all strings ("proofs") onto "theorems" (elements of certain language L; if L is the language of all propositional tautologies, the system is called a *propositional* proof system). The existence of a *polynomially bounded* propositional proof system (that is, a system that has a polynomial-size proof for every tautology) is equivalent to $\mathbf{NP} = \mathbf{co\text{-}NP}$. In the context of polynomial boundedness a proof system can be equivalently viewed as a function that given a formula and a "proof", verifies in polynomial time that a formula is a tautology: it must accept at least one "proof" for each tautology (*completeness*) and reject all proofs for non-tautologies (*soundness*).

One proof system Π_w is *simulated* by another one Π_s if the shortest proofs for every tautology in Π_s are at most polynomially longer than the shortest proofs in Π_w. The notion of *p-simulation* is similar, but requires also a polynomial-time computable function for translating the proofs from Π_w to Π_s. A *(p-)optimal* propositional proof system is one that (p-)simulates all other propositional proof systems.

The existence of an optimal (or p-optimal) propositional proof system is a major open question. If one would exist, it would allow to reduce the **NP** vs **co-NP** question to proving proof size bounds for just one proof system. It would also imply the existence of a complete disjoint **NP** pair [Raz94, Pud03]. Krajíček and Pudlák [KP89] show that the existence of a p-optimal system is equivalent to the existence of an algorithm that is optimal on all propositional tautologies, namely, it always solves the problem correctly and it takes for it at most polynomially longer to stop on every tautology than for any other correct algorithm *on the same tautology*. Monroe [Mon09] recently gave a conjecture implying that such algorithm does not exist. Note that Levin [Lev73] showed that an optimal algorithm does exist for finding witnesses to non-tautologies; however, (1) its behaviour on tautologies is not restricted; (2) after translating to the decision problem by self-reducibility the running time in the optimality condition is compared to the running time for *all shorter formulas as well*.

An *automatizable* proof system is one that has an automatization procedure that given a tautology, outputs its proof of length polynomially bounded by the length of the shortest proof in time bounded by a polynomial in the output length. The automatizability of a proof system Π implies polynomial separability of its canonical **NP** pair [Pud03], and the latter implies the automatizability of a system that p-simulates Π. This, however, does not imply the existence of (p-)optimal propositional proof systems in the class of automatizable proof systems. To the best of our knowledge, no such system is known to the date.

Proving propositional tautologies heuristically. An obvious obstacle to constructing an optimal proof system by enumeration is that no efficient procedure is known for enumerating the set of all complete and sound proof systems. Recently a number of papers overcome similar obstacles in other settings by considering either computations with non-uniform advice (see [FS06] for survey) or *heuristic* algorithms [FS04, Per07, Its09]. In particular, optimal propositional proof systems with advice do exist [CK07]. We try to follow the approach of heuristic computations to obtain a "heuristic" proof system. While our work is motivated by propositional proof complexity, i.e., proof systems for the set of propositional tautologies, our results apply to proof systems for any recursively enumerable language.

We introduce a notion of a *randomized heuristic automatizer* (a randomized semidecision procedure that may have false positives) and a corresponding notion of a *simulation*.

Its particular case, a deterministic automatizer (making no errors) for language L, along with deterministic simulations, can be viewed in two ways:

- as an automatizable proof system for L (note that such proof system can be identified with its automatization procedure; however, it may not be the case for randomized algorithms, whose running time may depend on the random coins), where simulations are p-simulations of proof systems;
- as an algorithm for L, where simulations are simulations of algorithms for L in the sense of [KP89].

Given $x \in L$, an automatizer must return 1 and stop. The question (handled by simulations) is how fast it does the job. For $x \notin L$, the running time does *not* matter. Given $x \notin L$, a deterministic automatizer simply must *not* return 1. A randomized heuristic automatizer may erroneously return 1; however, for "most" inputs it may do it only with bounded probability ("good" inputs). The precise notion of "most" inputs is: given an integer parameter d and a sampler for $\overline{L}$, "bad" inputs must have probability less than $1/d$ according to the sampler. The parameter d is handled by simulations in the way such that no automatizer can stop in time polynomial in d and the length of input unless an optimal automatizer can do that.

In Sect. 2 we give precise definitions. In Sect. 3 we construct an optimal randomized heuristic automatizer. In Sect. 4 we give a notion of heuristic probabilistic proof system and discuss the relation of automatizers to such proof systems.

2. Preliminaries

2.1. Distributional proving problems

In this paper we consider algorithms and proof systems *that allow small errors*, i.e., claim a small amount of wrong theorems. Formally, we have a probability distribution concentrated on non-theorems and require that the probability of sampling a non-theorem accepted by an algorithm or validated by the system is small.

Definition 2.1. We call a pair (D, L) a *distributional proving problem* if D is a collection of probability distributions D_n concentrated on $\overline{L} \cap \{0,1\}^n$.

In what follows we write $\Pr_{x \leftarrow D_n}$ to denote the probability taken over x from such distribution, while $\Pr_A$ denotes the probability taken over internal random coins used by algorithm A.

2.2. Automatizers

Definition 2.2. A (λ, ϵ)-*correct automatizer* for distributional proving problem (D, L) is a randomized algorithm A with two parameters $x \in \{0,1\}^*$ and $d \in \mathbb{N}$ that satisfies the following conditions:

(1) A either outputs 1 (denoted $A(\ldots) = 1$) or does not halt at all (denoted $A(\ldots) = \infty$);
(2) For every $x \in L$ and $d \in \mathbb{N}$, $A(x, d) = 1$.
(3) For every $n, d \in \mathbb{N}$,

$$\Pr_{r \leftarrow D_n} \left\{ \Pr_A \{A(r, d) = 1\} > \epsilon \right\} < \frac{1}{\lambda d}.$$

Here $\lambda > 0$ is a constant and $\epsilon > 0$ may depend on the first input (x) length. An *automatizer* is a $(1, \frac{1}{4})$-correct automatizer.

Remark 2.3. For recursively enumerable L, conditions 1 and 2 can be easily enforced at the cost of a slight overhead in time by running L's semidecision procedure in parallel.

In what follows, all automatizers are for the same problem (D, L).

Definition 2.4. The *time* spent by automatizer A on input (x, d) is defined as the median time

$$t_A(x,d) = \min\left\{ t \in \mathbb{N} \;\middle|\; \Pr_A\{A(x,d) \text{ stops in time at most } t\} \geq \frac{1}{2} \right\}.$$

We will also use a similar notation for "probability p time":

$$t_A^{(p)}(x,d) = \min\left\{ t \in \mathbb{N} \;\middle|\; \Pr_A\{A(x,d) \text{ stops in time at most } t\} \geq p \right\}.$$

Definition 2.5. Automatizer S simulates automatizer W if there are polynomials p and q such that for every $x \in L$ and $d \in \mathbb{N}$,

$$t_S(x,d) \leq \max_{d' \leq q(d\cdot|x|)} p(t_W(x,d') \cdot |x| \cdot d).$$

Definition 2.6. An *optimal* automatizer is one that simulates every other automatizer.

Definition 2.7. Automatizer A is *polynomially bounded* if there is a polynomial p such that for every $x \in L$ and every $d \in \mathbb{N}$,

$$t_A(x,d) \leq p(d \cdot |x|).$$

The following proposition follows directly from the definitions.

Proposition 2.8.

(1) *If W is polynomially bounded and is simulated by S, then S is polynomially bounded too.*
(2) *An optimal automatizer is not polynomially bounded if and only if no automatizer is polynomially bounded.*

3. Optimal automatizer

The optimal automatizer that we construct runs all automatizers in parallel and stops when the first of them stops (recall Levin's optimal algorithm for SAT [Lev73]). A major obstacle to this simple plan is the fact that it is unclear how to enumerate all automatizers efficiently (put another way, how to check whether a given algorithm is a correct automatizer). The plan of overcoming this obstacle (similar to constructing a complete public-key cryptosystem [HKN$^+$05] (see also [GHP09])) is as follows:

- Prove that w.l.o.g. a correct automatizer is very good: in particular, amplify its probability of success.
- Devise a "certification" procedure that distinguishes very good automatizers from incorrect automatizers with overwhelming probability.
- Run all automatizers in parallel, try to certify automatizers that stop, and halt when the first automatizer passes the check.

The amplification is obtained by repeating and the use of Chernoff bounds.

Proposition 3.1 (Chernoff bounds (see, e.g., [MR95, Chapter 4]))**.**
Let $X_1, X_2, \ldots, X_n \in \{0,1\}$ be independent random variables. Then if X is the sum of X_i and if μ is $\mathbf{E}[X]$, for any δ, $0 < \delta \le 1$:

$$\Pr\{X < (1-\delta)\mu\} < e^{-\mu\delta^2/2}, \qquad \Pr\{X > (1+\delta)\mu\} < e^{-\mu\delta^2/3}.$$

Corollary 3.2. *Let $X_1, X_2, \ldots, X_n \in \{0,1\}$ be independent random variables. Then if X is the sum of X_i and if $1 \ge \mu_1 \ge \mathbf{E}[X] \ge \mu_2 \ge 0$, for any δ, $0 < \delta \le 1$:*

$$\Pr\{X < (1-\delta)\mu_2\} < e^{-\mu_2\delta^2/2}, \qquad \Pr\{X > (1+\delta)\mu_1\} < e^{-\mu_1\delta^2/3}.$$

Lemma 3.3 (amplification)**.** *Every automatizer W is simulated by a $(4, e^{-m/48})$-correct automatizer S, where $m \in \mathbb{N}$ may depend at most polynomially on $d \cdot |x|$ (for input (x,d)). Moreover, there are polynomials p and q such that for every $x \in L$ and $d \in \mathbb{N}$,*

$$t_S^{(1-e^{-m/64})}(x,d) \le \max_{d' \le q(d\cdot|x|)} p(t_W(x,d')). \tag{3.1}$$

Proof. $S(x,d)$ runs m copies of $W(x,4d)$ in parallel and stops as soon as the $\frac{3}{8}$ fraction of copies stop.

By Chernoff bounds, S is $(4, e^{-m/48})$-correct. The "strong" simulation condition (3.1) is satisfied because by Chernoff bounds the running time of the fastest $\frac{3}{8}$ fraction of executions is less than median time with probability at least $1 - e^{-m/64}$. ∎

Theorem 3.4 (optimal automatizer)**.** *Let (D,L) be a distributional proving problem, where L is recursively enumerable and D is polynomial-time samplable, i.e., there is a polynomial-time randomized Turing machine that given 1^n on input outputs x with probability $D_n(x)$ for every $x \in \{0,1\}^n$. Then there exists an optimal automatizer for (D,L).*

Proof. For algorithm A, we say that it is (λ,ϵ)*-correct for input length n and parameter d* if it it satisfies condition 3 of Definition 2.2 for n and d. If an algorithm is (λ,ϵ)-correct for every n (resp., every d), we omit n (resp., d).

In order to check an algorithm for correctness, we define a *certification* procedure that takes an algorithm A and distinguishes between the cases where A is $(4, \frac{1}{18d\log_*^2 n})$-correct for given n, d (from Lemma 3.3 we know that one can assume such correctness) or it is not $(1, \frac{1}{16d\log_*^2 n})$-correct ($(1, \frac{1}{16d\log_*^2 n})$-correct automatizers suffice for the correctness of further constructions). W.l.o.g. we may assume that

$$A \text{ satisfies conditions 1 and 2 of Definition 2.2} \tag{3.2}$$

(for the latter condition, notice that L is recursively enumerable and one may run its semidecision procedure in parallel).

The certification procedure has a subroutine TEST that estimates the probability of A's error simply by repeating A and couting its faults.

TEST(A, x, d', T, l, f):

(1) Repeat for each $i \in \{1, \ldots, l\}$
  (a) If $A(x,d')$ stops in T steps, let $c_i = 1$; otherwise let $c_i = 0$.
(2) If $\sum_i c_i \ge l/f$, then reject; otherwise accept.

Lemma 3.5. *For every A, x, d', T, l, f,*

(1) *If $A(x,d')$ stops with probability less than $\frac{1}{1.01f}$, then* TEST *will reject it with probability less than $e^{-\frac{l}{3.03\cdot 10^4\cdot f}}$.*
(2) *If $A(x,d')$ stops in time at most T with probability more than $\frac{1}{0.99f}$, then* TEST *will accept it with probability less than $e^{-\frac{l}{2\cdot 10^4\cdot f}}$.*

Proof. Follows directly from Chernoff bounds. ∎

CERTIFY(A,n,d',T,k,l,f):

(1) Repeat for each $i\in\{1,\dots,k\}$
  (a) Generate x_i according to D_n.
  (b) If TEST(A,x_i,d',T,l,f) rejects, let $b_i=1$; otherwise let $b_i=0$.
(2) If $\sum_i b_i \geq k/(2d')$, then reject; otherwise accept.

Lemma 3.6. *Let $d,n,T\in\mathbb{N}$. Let A be an algorithm pretending to be an automatizer. Run*

$$\text{CERTIFY}(A,n,d',T,k,l,f).$$

Then

(1) *If A is $(4,\frac{1}{1.011f})$-correct, then A is accepted by* CERTIFY *almost for sure, failing with probability less than $e^{-\frac{k}{12d'}}+k\cdot e^{-\frac{l}{3.03\cdot 10^4\cdot f}}$.*
(2) *Let A^T be a restricted version of A that behaves similarly to A for T steps and enters an infinite loop afterwards. If A^T is not $(1,\frac{1}{0.99f})$-correct for length n and parameter d, then A is accepted by* CERTIFY *with probability less than $e^{-\frac{k}{8d'}}+k\cdot e^{-\frac{l}{2\cdot 10^4\cdot f}}$.*

Proof. 1. Let $\Delta=\{x\in\operatorname{Im} D_n \mid \Pr\{A(x,d)=1\}>\frac{1}{1.011f}\}$. By assumption, $D_n(\Delta)<\frac{1}{4d'}$.

The certification procedure takes k samples from D_n. For every sample $x_i\in\overline{L}\setminus\Delta$, the probability that the corresponding b_i equals 1 is less than $e^{-\frac{l}{3.03\cdot 10^4\cdot f}}$. Thus, the probability that there is a sample x_i from $\overline{L}\setminus\Delta$ that yields $b_i=1$ is less than $k\cdot e^{-\frac{l}{3.03\cdot 10^4\cdot f}}$. Denote this unfortunate event by E. If it does not hold, only samples from Δ may cause $b_i=1$ and by Chernoff's bound

$$\Pr\{\sum_i b_i \geq k/(2d') \mid \overline{E}\} < e^{-\frac{k}{12d'}}.$$

Thus, the total probability of reject is as claimed.

2. Let $\Delta=\{x\in\operatorname{Im} D_n \mid \Pr\{A(x,d)=1\}>\frac{1}{0.99f}\}$. By assumption, $D_n(\Delta)\geq\frac{1}{d'}$.

The certification procedure takes k samples from D_n. For every sample $x_i\in\Delta$, the probability that the corresponding b_i equals 0 is less than $e^{-\frac{l}{2\cdot 10^4\cdot f}}$. Thus, the probability that there is a sample x_i from Δ that yields $b_i=0$ is less than $k\cdot e^{-\frac{l}{2\cdot 10^4\cdot f}}$. Denote this unfortunate event by E. Assuming it does not hold only samples outside Δ may cause $b_i=0$ and by Chernoff's bound

$$\Pr\{\sum_i b_i < k/(2d') \mid \overline{E}\} < e^{-\frac{k}{8d'}}.$$

∎

We now define the optimal automatizer U. It works as follows:

$U(x, d)$:

(1) Let

$$\begin{aligned} n &= |x|, \\ d' &= 16d \log_*^2 n, \\ f &= 17d \log_*^2 n, \\ k &= 12d' \ln(16d \log_*^2 n), \\ l &= (3.03 \cdot 10^4) \cdot f \cdot \ln(16kd \log_*^2 n). \end{aligned}$$

(2) Run the following processes for $i \in \{1, \ldots, \log_* n\}$ in parallel:
   (a) Run $A_i(x, d')$, the algorithm with Turing number i satisfying assumption (3.2), and compute the number of steps T_i made by it before it stops.
   (b) If CERTIFY$(A_i, n, d', T_i, k, l, f)$ accepts, then output 1 and stop U (all processes).

(3) If none of the processes has stopped, go into an infinite loop.

Correctness. We now show that U errs with probability less than 1/4.

What are the inputs that cause U to error? For every such input x there exists $i \le \log_* n$ such that

$$u_x^i = \sum_{T=1}^{\infty} p_{x,T}^i c_T^i \ge \frac{1}{4 \log_* n}, \tag{3.3}$$

where

$$\begin{aligned} p_{x,t}^i &= \Pr\{A_i(x, d') \text{ stops in exactly } t \text{ steps}\}, \\ c_t^i &= \Pr\{\text{CERTIFY}(A_i, n, d', t, k, l, f) \text{ accepts}\}. \end{aligned}$$

Let E_i be the set of inputs $x \notin L$ satisfying inequality (3.3).

We claim that $D(E_i) < \frac{1}{d \log_* n}$, which suffices to show the $(1, 1/4)$-correctness.

Assume the contrary. Let $T_i = \min\{t \mid c_t^i < e^{-\frac{k}{8d'}} + k \cdot e^{-\frac{l}{2 \cdot 10^4 \cdot f}}\}$. Note that by Lemma 3.6 $A_i^{T_i - 1}$ is $(1, \frac{1}{0.99f})$-correct for n and d', i.e.,

$$\Pr_{x \leftarrow D_n}\{\sum_{T < T_*^i} p_{x,T}^i > \frac{1}{0.99f}\} < \frac{1}{d'}.$$

We omit i and n in the estimations that follow. Here is how we get a contradiction:

$$\frac{1}{4d\log_*^2 n} \le \frac{D(E_i)}{4\log_* n} = \sum_{x\in E_i} \frac{1}{4\log_* n} D(x) \le \sum_{x\in E_i} u_x D(x) \le$$

$$\sum_{x\notin L} u_x D(x) = \sum_{x\notin L} \sum_{T=1}^{\infty} p_{x,T} c_T D(x) =$$

$$\sum_{x\notin L} \left(\sum_{T<T_*} p_{x,T} c_T D(x) + \sum_{T\ge T_*} p_{x,T} c_T D(x) \right) \le$$

$$\sum_{T<T_*} \left(\sum_{x\notin L,\ \sum_{t<T_*} p_{x,t} \le \frac{1}{0.99f}} p_{x,T} D(x) + \sum_{x\notin L,\ \sum_{t<T_*} p_{x,t} > \frac{1}{0.99f}} p_{x,T} D(x) \right)$$

$$+ e^{-\frac{k}{8d'}} + k\cdot e^{-\frac{l}{2\cdot 10^4\cdot f}} \le$$

$$\frac{1}{0.99f} + \frac{1}{d'} + e^{-\frac{k}{8d'}} + k\cdot e^{-\frac{l}{2\cdot 10^4\cdot f}} < \frac{1}{16d\log_*^2 n} + \frac{1}{16d\log_*^2 n} + \frac{1}{8d\log_*^2 n} = \frac{1}{4d\log_*^2 n}.$$

Simulation. Assume we are give a correct automatizer A^s. Plug in $m = 48\cdot\ln(18d\log_*^2 n)$ into Lemma 3.3. The lemma yields that A^s is "strongly" simulated by a $(4, \frac{1}{18d\log_*^2 n})$-correct automatizer A. It remains to estimate, for given "theorem" $x \in L$, the (median) running time of U in terms of $t_A^{(1-e^{-m/64})}(x,d) = t_A^{(1-\frac{1}{(18d\log_*^2 n)^{3/4}})}(x,d)$ (as we know that the latter is bounded by $\max_{d'\le q(d\cdot|x|)} p(t_{A^s}(x,d'))$ for a polynomials p and q).

Since the definition of simulation is asymptotic, we consider only x of length greater than the Turing number of A. By Lemma 3.6, A is not certified with probability less than $e^{-\frac{k}{12d'}} + k\cdot e^{-\frac{l}{3.03\cdot 10^4\cdot f}} \le \frac{1}{8d\log_*^2 n}$. If A is certified, U stops in time upper bounded by a polynomial of the time spent by A with an overhead polynomial in $|x|$ and d for running other algorithms and the certification procedures. Thus the median time $t_U(x,d)$ is bounded by a polynomial in $|x|$, d, and $t_A^{(\frac{1}{2}+\frac{1}{8d\log_*^2 n})}(x,d) \le t_A^{(1-\frac{1}{(18d\log_*^2 n)^{3/4}})}(x,d)$. ∎

4. Heuristic proof systems

In this section we define proof systems that make errors (claim a small fraction of wrong theorems). We consider automatizable systems of this kind and show that every such system defines an automatizer taking time at most polynomially larger than the length of the shortest proof in the initial system. This shows that automatizers form a more general notion than automatizable heuristic proof systems. The opposite direction is left as an open question.

Definition 4.1. Randomized Turing machine Π is a *heuristic proof system* for distributional proving problem (D, L) if it satisfies the following conditions.

(1) The running time of $\Pi(x, w, d)$ is bounded by a polynomial in d, $|x|$, and $|w|$.

(2) (Completeness) For every $x \in L$ and every $d \in \mathbb{N}$, there exists a string w such that $\Pr\{\Pi(x, w, d) = 1\} \geq \frac{1}{2}$. Every such string w is called a $\Pi^{(d)}$-proof of x.
(3) (Soundness) $\Pr_{x \leftarrow D_n}\{\exists w : \Pr\{\Pi(x, w, d) = 1\} > \frac{1}{4}\} < \frac{1}{d}$.

Definition 4.2. Heuristic proof system is *automatizable* if there is a randomized Turing machine A satisfying the following conditions.

(1) For every $x \in L$ and every $d \in \mathbb{N}$, with probability at least $\frac{1}{2}$ algorithm $A(x, d)$ outputs a correct $\Pi^{(d)}$-proof of size bounded by a polynomial in d, $|x|$, and $|w|$, where w is the shortest $\Pi^{(d)}$-proof of x.
(2) The running time of $A(x, d)$ is bounded by a polynomial in $|x|$, d, and the size of its own output.

Definition 4.3. We say that heuristic proof system Π_1 *simulates* heuristic proof system Π_2 if there exist polynomials p and q such that for every $x \in L$, the shortest $\Pi_1^{(d)}$-proof of x has size at most

$$p(d \cdot |x| \cdot \max_{d' \leq q(|x|d)} \{\text{the size of the shortest } \Pi_2^{(d')}\text{-proof of } x\}).$$

Note that this definition essentially ignores proof systems that have much shorter proofs for some inputs than the inputs themselves. We state it this way for its similarity to the automatizers case.

Definition 4.4. Heuristic proof system Π is *polynomially bounded* if there exists a polynomial p such that for every $x \in L$ and every $d \in \mathbb{N}$, the size of the shortest $\Pi^{(d)}$-proof of x is bounded by $p(|x|d)$.

Proposition 4.5. *If heuristic proof system Π_1 simulates system Π_2 and Π_2 is polynomially bounded, then Π_1 is also polynomially bounded.*

We now show how automatizers and automatizable heuristic proof systems are related.

Consider automatizable proof system (Π, A) for distributional proving problem (D, L) with recursively enumerable language L. Let us consider the following algorithm $A_\Pi(x, d)$:

(1) Execute 1000 copies of $A(x, d)$ in parallel.
For each copy,
(a) if it stops with result w, then
- execute $\Pi(x, w, d)$ 10000 times;
- if there were at least 4000 accepts of Π (out of 10000), stop all parallel processes and output 1.

(2) Execute the enumeration algorithm for L; output 1 if this algorithm says that $x \in L$; go into an infinite loop otherwise.

Proposition 4.6. *If (Π, A) is a (correct) heuristic automatizable proof system for recursively enumerable language L, then A_Π is a correct automatizer for $x \in L$ and $t_{A_\Pi}(x, d)$ is bounded by polynomial in size of the shortest Π_d-proof of x.*

Proof. Soundness (condition 3 in Def. 2.2). Let $\Delta_n = \{x \in \overline{L} \mid \exists w : \Pr\{\Pi(x, w, d) = 1\} > \frac{1}{4}\}$. By definition, $D_n(\Delta_n) < \frac{1}{d}$. For $x \in \{0,1\}^n \setminus \Delta_n$ and specific w, Chernoff bounds imply that $\Pi(x, w, d)$ accepts in 0.4 or more fraction of executions with exponentially small probability, which remains much smaller than $\frac{1}{4}$ even after multiplying by 1000.

Completeness (conditions 2 and 1 in Def. 2.2) is guaranteed by the execution of the semi-decision procedure for L.

Simulation. For $x \in L$, the probability that A errs 1000 times is negligible (at most 2^{-1000}). Thus with high probability at least one of the parallel executions of $A(x,d)$ outputs a correct Π_d-proof of size bounded by a polynomial in the size of the shortest Π_d-proof of x. For $x \in L$ and (correct) $\Pi^{(d)}$-proof w, Chernoff bounds imply that $\Pi(x,w,d)$ accepts in at least 0.4 fraction of executions with probability close to 1. Therefore, $t_{A_\Pi}(x,d)$ is bounded by a polynomial in $|x|$, d, and the size of the shortest Π_d-proof of x. ∎

5. Further research

One possible direction is to show that automatizers are equivalent to automatizable heuristic proof systems or, at least, that there is an optimal automatizable heuristic proof system. That may require some tweak in the definitions, because the first obstacle to proving the latter fact is the inability to check a candidate proof system for the non-existence of a much shorter (correct) proof than those output by a candidate automatizer.

Also Krajíček and Pudlák [KP89] and Messner [Mes99] list equivalent conditions for the existence of (deterministic) optimal and p-optimal proof systems. It seems promising (and, in some places, challenging) to prove similar statements in the heuristic setting.

Acknowledgements

During the work on the subject, we discussed it with many people. Our particular thanks go to (in the alphabetical order) Dima Antipov, Dima Grigoriev, and Sasha Smal.

References

[CK07] Stephen A. Cook and Jan Krajíček. Consequences of the provability of $NP \subseteq P/poly$. *The Journal of Symbolic Logic*, 72(4):1353–1371, 2007.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, March 1979.

[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.

[FS06] Lance Fortnow and Rahul Santhanam. Recent work on hierarchies for semantic classes. *SIGACT News*, 37(3):36–54, 2006.

[GHP09] Dima Grigoriev, Edward A. Hirsch, and Konstantin Pervyshev. A complete public-key cryptosystem. *Groups, Complexity, Cryptology*, 1(1):1–12, 2009.

[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.

[Its09] Dmitry M. Itsykson. Structural complexity of AvgBPP. In *Proceedings of 4th International Computer Science Symposium in Russia*, volume 5675 of *Lecture Notes in Computer Science*, pages 155–166, 2009.

[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.

[Lev73] Leonid A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9:265–266, 1973.

[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.

[Mon09] Hunter Monroe. Speedup for natural problems and coNP?=NP. Technical Report 09-056, Electronic Colloquium on Computational Complexity, 2009.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.

[Per07] Konstantin Pervyshev. On heuristic time hierarchies. In *Proceedings of the 22nd IEEE Conference on Computational Complexity*, pages 347–358, 2007.

[Pud03] Pavel Pudlák. On reducibility and symmetry of disjoint NP pairs. *Theoretical Computer Science*, 295(1–3):323–339, 2003.

[Raz94] Alexander A. Razborov. On provably disjoint NP-pairs. Technical Report 94-006, Electronic Colloquium on Computational Complexity, 1994.