On optimal heuristic randomized semidecision procedures, with applications to proof complexity and cryptography*

Edward A. Hirsch Dmitry Itsykson Ivan Monakhov
Alexander Smal
Steklov Institute of Mathematics at St. Petersburg

December 5, 2010

Abstract

The existence of an 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 on all propositional tautologies. Monroe [Mon09] recently presented a conjecture implying that such an algorithm does not exist.

We show that if one allows errors, then such optimal algorithms *do* exist. The concept is motivated by the notion of heuristic algorithms. Namely, we allow an algorithm, called a *heuristic acceptor*, to claim a small number of false "theorems" and err with bounded probability on other inputs. The amount of false "theorems" is measured according to a polynomial-time samplable distribution on non-tautologies. Our result remains valid for all recursively enumerable languages and can also be viewed as the existence of an optimal weakly automatizable heuristic proof system. The notion of a heuristic acceptor extends the notion of a classical acceptor; in particular, an optimal heuristic acceptor for any distribution simulates every classical acceptor for the same language.

We also note that the existence of a **co-NP**-language L with a polynomial-time samplable distribution on $\overline{L}$ that has no polynomial-time heuristic acceptors is equivalent to the existence of an infinitely-often one-way function.

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 the 1970s,

*Supported by the Federal Target Programme "Scientific and scientific-pedagogical personnel of the innovative Russia" 2009–2013, by the Programme of Fundamental Research of RAS, and by the President of Russia grant MK-4089.2010.1. Work was done while the third author was a master student in St. Petersburg Academic University supported by Yandex Fellowship.

these important questions are still open for most interesting languages, for example, the language of propositional tautologies.

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", i.e., elements of some language L. If L is the language of all propositional tautologies, the system is called a *propositional* proof system. A *polynomially bounded* propositional proof system is a system that has a polynomial-size proof for every word x in L. The existence of a polynomially bounded propositional proof system is equivalent to **NP** = **co-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*).

A proof system Π_w is *simulated* by a proof system 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 existed, it would allow one 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] showed 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 accepts every tautology at most polynomially slower than any other correct algorithm does, when run *on the same tautology*. Then Sadowski [Sad99] proved a similar equivalence for `SAT`. Finally, Messner [Mes99] gave a different proof of these equivalences extending them to many other languages. He also coined the term "optimal acceptors" for such algorithms. Monroe [Mon09] recently presented a conjecture implying that optimal acceptors for `TAUT` do 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) decision algorithms can in general work faster than algorithms that output witnesses. If one applies the well-known `SAT` search-to-decision reduction, one may get a different formula on which the decision algorithm may run much slower than on the original formula.

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. All systems simulated by an automatizable proof system are called *weakly automatizable*. The automatizability of a proof system Π implies the polynomial separability of its canonical **NP** pair [Pud03], and the latter implies the weak automatizability of Π. This, however, does not imply the existence of (p-)optimal propositional proof systems in the class of automatizable or weakly automatizable proof systems. To the best of our knowledge, no such system is known to the date.

Heuristic approach to checking propositional tautologies. 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 have overcome similar obstacles in other settings by considering either computations with non-uniform advice (see [FS06] for a survey) or *heuristic* algorithms [FS04, Per07, Its10]. 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.

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

However, first of all, to formalize the notion of "a small number of false positives", we need to introduce a new concept of a computational problem, a *distributional proving problem* (D, L) which consists of a language L and a polynomial-time samplable distribution on $\overline{L}$. The difference from the average-case complexity notion of a distributed problem is that D is concentrated on $\overline{L}$. For example, L can be the language of unsatisfiable formulas in 3-CNF, and D can be a "planted" `SAT` distribution that selects uniformly at random a certain number of 3-clauses satisfied by one or more fixed ("planted") assignments.

A heuristic acceptor must accept every word in L. Its behaviour on the complement of L splits the complement into two subsets:

- "good" inputs: these are the inputs that are accepted with small probability bounded from the above; the probability is taken over the internal random coins of the acceptor;
- "bad" inputs: here the acceptor may err (i.e., accept) with unbounded probability.

"Bad" inputs must have D-measure at most $1/d$, where d is an additional integer parameter given to the acceptor on its input. (Naturally, it is possible that the acceptor takes more time for larger values of d.)

The notion of simulation does *not* depend on the distribution D. A heuristic acceptor A simulates a heuristic acceptor B if for every x in L, the time taken by $A(x, d)$ is bounded by a polynomial in d, $|x|$, and the time taken by $B(x, d')$ for the values of d' that are polynomially close to d. An optimal randomized heuristic acceptor is the "fastest" acceptor, i.e., the one that simulates every randomized heuristic acceptor. We prove that an optimal randomized heuristic acceptor exists. Since the notion of a heuristic acceptor extends the notion of a classical acceptor, an optimal heuristic acceptor for any distribution also classically simulates every classical acceptor for the same language.

Polynomially bounded heuristic acceptors. Similarly to the classical case, the notion of an optimal heuristic acceptor makes sense only for languages and samplers that have no polynomially bounded heuristic acceptors, i.e., acceptors working in time polynomial in the size of the input and the parameter d. It turns out that such polynomial-time samplers and languages in **co-NP** roughly correspond to pseudo-random generators and the complements of their images, respectively (recall the suggestion of [ABSRW00, Kra01a, Kra01b] to consider such problems for proving lower bounds for classical proof

systems). More precisely, such an intractable pair exists if and only if there is an infinitely-often one-way function.

Relation to proof systems. We also define a proof-complexity counterpart of randomized heuristic acceptors: randomized heuristic proof systems. These systems have randomized verifiers and accept proofs of a small number of "non-theorems" (we postpone precise definitions to Sect. 4).

As said above, in the classical case optimal acceptors exist if and only if optimal proof systems exist. We are currently unable to prove such an equivalence in the heuristic case. It is not even immediately obvious that the equivalence to weakly automatizable and automatizable heuristic proof systems, trivial in the classical case, holds for heuristic acceptors. We prove that it is indeed the case that heuristic acceptors are equivalent to weakly automatizable heuristic proof systems, which gives an optimal weakly automatizable heuristic proof system.

In Sect. 2 we give precise definitions. In Sect. 3 we construct an optimal randomized heuristic acceptor. In Sect. 4 we introduce the notion of a randomized heuristic proof system and show that weakly automatizable heuristic proof systems are equivalent to heuristic acceptors. In Sec. 5 we show that the existence of problems intractable for heuristic acceptors is equivalent to the existence of infinitely-often one-way functions. We also provide a complete distributional proving problem. Finally, in Sect. 6 we list possible directions for further research.

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 a 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 $\overline{L} \cap \{0,1\}^n$.

A distribution D is *polynomial-time samplable* if there is a polynomial-time randomized Turing machine (*sampler*) that, given an input 1^n, outputs x with probability $D_n(x)$ for every $x \in \{0,1\}^n$. In this paper we consider distributional proving problems with polynomial-time samplable distributions.

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

2.2 Heuristic acceptors

A heuristic acceptor is an algorithm that always accepts inputs in L; for "most" inputs not in L, it may accept with a small probability only, and for other inputs not in L it

may err arbitrarily. In what follows we will be interested in the running time of such an acceptor on inputs in L, but not on inputs not in L.

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

1. A either accepts, i.e., outputs 1 (denoted $A(\ldots) = 1$), or does not halt at all.
2. For every $x \in L$ and $d \in \mathbb{N}$, $\Pr_A\{A(x,d) = 1\} = 1$.
3. For every $n, d \in \mathbb{N}$,
$$\Pr_{r \leftarrow D_n}\left\{\Pr_A\{A(r,d) = 1\} \geq \frac{1}{8}\right\} < \frac{1}{d}.$$

A *normalized heuristic acceptor* is defined similarly, but condition (3) is replaced by

3′. For every $n, d \in \mathbb{N}$,
$$\Pr_{\substack{r \leftarrow D_n \\ A}}\{A(r,d) = 1\} < \frac{1}{d}.$$

Remark 2.1. A normalized heuristic acceptor is *not* necessarily a heuristic acceptor for the same input. However, it is a heuristic acceptor for a different value of d, as we will show below.

Remark 2.2. For every recursively enumerable language L, a semidecision procedure is a heuristic acceptor for arbitrary D.

The time spent by a heuristic acceptor may depend on its random coins. Therefore the main complexity characteristic of a heuristic acceptor is its median time.

Definition 2.3. The median running time of an algorithm A on an input z is
$$t_A(z) = \min\{t \mid \Pr_A\{A(z) \text{ stops in at most } t \text{ steps}\} \geq \frac{1}{2}\}.$$

We will also use a similar notation for "probability p time":
$$t_A^{(p)}(z) = \min\{t \mid \Pr_A\{A(z) \text{ stops in at most } t \text{ steps}\} \geq p\}.$$

Definition 2.4. A function $f : \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ is *polynomially bounded on a language* L if there is a polynomial p such that for all $x \in L$ and $d \in \mathbb{N}$, $f(x,d) \leq p(|x|d)$.

Definition 2.5. A function $f : \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ *dominates* a function $g : \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ on a language L (denoted $f \succeq g$) if there are polynomials p and q such that for all $x \in L$ and $d \in \mathbb{N}$,
$$g(x,d) \leq \max_{d' \leq q(|x|d)}\{p(f(x,d')d|x|)\}.$$

Remark 2.3.

1. If $f \succeq g$ on L and f is polynomially bounded on L, then so is g.

2. the relation $\succeq$ is transitive.

Definition 2.6. A heuristic acceptor A for a distributional proving problem (D, L) is *polynomially bounded* if $t_A(x, d)$ is polynomially bounded on L.

Definition 2.7. Given heuristic acceptors A and A' for the same language L, the heuristic acceptor A *simulates* A' if $t_{A'} \succeq t_A$ on L.

A heuristic acceptor A *strongly simulates* A' if $t_{A'} \succeq t_A^{(3/4)}$ on L.

Proposition 2.1 (Chernoff–Hoeffding bound [McD98])**.** For independent random variables $X_i \in \{0, 1\}$, $1 \le i \le N$, with $\mathbf{E}\, X_i = \mu$ for every $\epsilon > 0$,

$$\Pr\left\{\left|\frac{\sum_{i=1}^N X_i}{N} - \mu\right| \ge \epsilon\right\} \le 2e^{-2\epsilon^2 N}.$$

Proposition 2.2. For every heuristic acceptor A for (D, L), there is a normalized heuristic acceptor B that strongly simulates A.

Proof. The new algorithm $B(x, d)$ runs ℓ instances of $A(x, 2d)$ independently in parallel (where ℓ is to be defined later) and accepts as soon as at least $\frac{5}{16}\ell$ instances of A stop. That many instances must stop in time at most $t_A(x, 2d)$ with probability at least $1 - 2e^{-\frac{9\ell}{128}}$ by the Chernoff–Hoeffding bound. Thus for every $x \in L$ and ℓ big enough, $t_B^{(3/4)}(x, d) \le \mathrm{poly}(\ell, t_A(x, 2d))$.

Let $x \in \operatorname{supp} D_n$. Split $\operatorname{supp} D_n$ into $S_1 = \{x \in \operatorname{supp} D_n \mid \Pr_A\{A(x, 2d) = 1\} < \frac{1}{8}\}$ and $S_2 = \operatorname{supp} D_n \setminus S_1$. Since A is a heuristic acceptor, $D_n(S_2) < \frac{1}{2d}$. For $x \in S_1$, the Chernoff–Hoeffding bound yields the exponentially small probability $2e^{-\frac{9\ell}{128}}$ of B accepting x. Choose ℓ such that $2e^{-\frac{9\ell}{128}} < \frac{1}{4d}$. Then $\Pr_{x \leftarrow D_n;\ B}\{B(x, d) = 1\} = \Pr_{x \leftarrow D_n;\ B}\{B(x, d) = 1 \mid x \in S_1\} D_n(S_1) + \Pr_{x \leftarrow D_n}\{B(x, d) = 1 \mid x \in S_2\} D_n(S_2) \le \Pr_{x \leftarrow D_n;\ B}\{B(x, d) = 1 \mid x \in S_1\} + D_n(S_2) < \frac{1}{4d} + \frac{1}{2d} < \frac{1}{d}$. □

Proposition 2.3. If $\Pr_{x \leftarrow D_n;\ A}\{A(x, d) = 1\} < \frac{1}{d}$ holds for every d, then for every $\lambda > 0$, $\Pr_{x \leftarrow D_n}\{\Pr_A\{A(x, \lambda d) = 1\} \ge \frac{1}{\lambda}\} < \frac{1}{d}$.

Proof. We have $\frac{1}{\lambda d} > \Pr_{x \leftarrow D_n; A}\{A(x, \lambda d) = 1\} \ge \Pr_{x \leftarrow D_n; A}\{A(x, \lambda d) = 1 \mid \Pr_A\{A(x, \lambda d) = 1\} \ge \frac{1}{\lambda}\} \cdot \Pr_{x \leftarrow D_n}\{\Pr_A\{A(x, \lambda d) = 1\} \ge \frac{1}{\lambda}\} \ge \frac{1}{\lambda} \Pr_{x \leftarrow D_n}\{\Pr_A\{A(x, \lambda d) = 1\} \ge \frac{1}{\lambda}\}$. □

Corollary 2.1. If A is a normalized heuristic acceptor, then $B(x, d) = A(x, 8d)$ is a heuristic acceptor simulating A.

Propositions 2.2 and 2.3 imply that every heuristic acceptor is simulated by a normalized heuristic acceptor, and vice versa.

3 Optimal heuristic acceptor

In this section, we construct an optimal heuristic acceptor, i.e., one that simulates every other heuristic acceptor. By Propositions 2.2 and 2.3 it suffices to construct an optimal normalized heuristic acceptor. Throughout the section, L is a recursively enumerable language.

The algorithm that we construct runs all heuristic acceptors 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 heuristic acceptors efficiently. Put another way, the problem is how to check whether a given algorithm is a correct heuristic acceptor. 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 heuristic acceptor is very good: in particular, amplify its probability of success.
- Devise a "certification" procedure that distinguishes very good heuristic acceptors from incorrect acceptors with overwhelming probability.
- Run all candidate heuristic acceptors in parallel, try to certify heuristic acceptors that stop, and halt when the first of them passes the check.

The certification procedure is as follows.

Algorithm 3.1 (Procedure $\textsc{Certify}(A, d', n, T, \ell, \delta)$)**.**

- Do ℓ times:
  - Sample $x \in D_n$.
  - Run $A(x, d')$ for at most T steps.
- Output 1, if at most $\delta\ell$ computations out of ℓ accepted.

Proposition 3.1. Let $A^{\leq T}$ be the algorithm that behaves as the algorithm A provided that A halts within the first T steps; otherwise, if A does not halt, $A^{\leq T}$ outputs $\perp$. Then

- If $\Pr_{x \leftarrow D_n;\ A}\{A^{\leq T}(x, d') = 1\} < \frac{\delta}{2}$, then $\Pr\{\textsc{Certify}(A, d', n, T, \ell, \delta) = 1\} \geq 1 - 2e^{-\delta^2\ell/2}$.
- If $\Pr_{x \leftarrow D_n;\ A}\{A^{\leq T}(x, d') = 1\} > 2\delta$, then $\Pr\{\textsc{Certify}\{A, d', n, T, \ell, \delta) = 1\} \leq 2e^{-2\delta^2\ell}$.

Proof. Follows from the Chernoff–Hoeffding bounds. □

We now construct an optimal normalized acceptor U.

Algorithm 3.2 (Algorithm $U(x, d)$)**.**

1. Run $A_1(x, 4dn), A_2(x, 4dn), \ldots, A_{\lfloor \frac{n}{2} \rfloor}(x, 4dn)$, and a semidecision procedure for L on input x in parallel, where $n = |x|$ and A_i is the algorithm with Goedel number i.
2. If A_i accepts in T_i steps, then run $\textsc{Certify}(A_i, 4dn, T_i, 2n^3d^3, \frac{1}{2dn})$. Accept if $\textsc{Certify}$ accepts, and otherwise terminate this parallel process without affecting other parallel processes.

3. Accept if the semidecision procedure accepts.

If one of the parallel threads accepts, all other processes are terminated.

Lemma 3.1. $U(x,d)$ is a normalized heuristic acceptor.

Proof. By construction U, either accepts or does not stop. For $x \in L$, it does stop because of the semidecision procedure for L.

Let τ_i denote $\max\{T \mid \Pr_{x\leftarrow D_n;\ A_i}\{A_i^{\leq T}(x,4dn)=1\} \leq \frac{1}{dn}\}$. Let $X_i(x)$ be the random variable equal to the number of steps that $A_i(x,4dn)$ makes before it accepts. If $A_i(x,4dn)$ does not accept, then $X_i(x)=\infty$. Let $C_i(T)$ denote the event $\textsc{Certify}(A_i,4dn,T,2n^3d^3,\frac{1}{2dn})=1$.

For every i we estimate the probability that U accepts on $\operatorname{supp} D$ because of A_i:

$$\begin{aligned}\Pr_{\substack{x\leftarrow D_n\\ A_i\\ \textsc{Certify}}}\{\exists T\in\mathbb{N} : X_i(x)=T\wedge C_i(T)\} &= \sum_{T=1}^{\infty}\Pr_{\substack{x\leftarrow D_n\\ A_i\\ \textsc{Certify}}}\{X_i(x)=T\wedge C_i(T)\}\\ &= \sum_{T=1}^{\tau_i}\Pr_{\substack{x\leftarrow D_n\\ A_i}}\{X_i(x)=T\}\Pr_{\textsc{Certify}}\{C_i(T)\} + \sum_{T>\tau_i}\Pr_{\substack{x\leftarrow D_n\\ A_i}}\{X_i(x)=T\}\Pr_{\textsc{Certify}}\{C_i(T)\}.\end{aligned} \tag{1}$$

We can estimate the first sum in (1) by the definition of τ_i:

$$\begin{aligned}\sum_{T=1}^{\tau_i}\Pr_{\substack{x\leftarrow D_n\\ A_i}}\{X_i(x)=T\}\Pr_{\textsc{Certify}}\{C_i(T)\} &\leq \sum_{T=1}^{\tau_i}\Pr_{\substack{x\leftarrow D_n\\ A_i}}\{X_i(x)=T\}\\ &= \Pr_{\substack{x\leftarrow D_n\\ A_i}}\{1\leq X_i(x)\leq\tau_i\} \leq \frac{1}{nd}.\end{aligned}$$

We can estimate the second sum in (1) using Proposition 3.1:

$$\begin{aligned}\sum_{T>\tau_i}\Pr_{\substack{x\leftarrow D_n\\ A_i}}\{X_i(x)=T\}\Pr_{\textsc{Certify}}\{C_i(T)\} &\leq \Pr_{\substack{x\leftarrow D_n\\ A_i}}\{X_i(x)>\tau_i\}\Pr_{\textsc{Certify}}\{C_i(\tau_i+1)\}\\ &\leq \Pr_{\textsc{Certify}}\{C_i(\tau_i+1)\} \overset{\text{Prop. 3.1}}{\leq} 2e^{-dn} < \frac{1}{dn}.\end{aligned}$$

Therefore, (1) is less than $\frac{2}{dn}$. The D-measure of the inputs that force U to erroneously output 1 is

$$\Pr_{\substack{x\leftarrow D_n\\ U}}\{U(x,d)=1\} \leq \sum_{i=1}^{\lfloor n/2\rfloor}\Pr_{\substack{x\leftarrow D_n\\ A_i\\ \textsc{Certify}}}\{\exists T\in\mathbb{N} : X_i(x)=T\wedge C_i(T)\} < \frac{n}{2}\cdot\frac{2}{dn} = \frac{1}{d}.$$

□

Lemma 3.2. The normalized heuristic acceptor U simulates every other normalized heuristic acceptor.

Proof. Let A be a heuristic acceptor. Then Proposition 2.2 yields a normalized heuristic acceptor B that strongly simulates A. Assume that B has Goedel number b. Then for $\lfloor \frac{n}{2} \rfloor > b$, it is run by U. By Proposition 3.1, CERTIFY accepts it with probability at least $1 - 2e^{-dn/2}$. Therefore, for $\lfloor \frac{n}{2} \rfloor > b$, with probability at least $\frac{1}{2}$ the running time of $U(x, d)$ is at most $p_1(dn \cdot t_B^{(3/4)}(x, 4dn))$ for some polynomial p_1, hence $t_U \preceq t_B^{(3/4)}$. Since B strongly simulates A, we have $t_B^{(3/4)} \preceq t_A$. Thus $t_U \preceq t_A$. □

Theorem 3.1. $U'(x, d) = U(x, 8d)$ is a heuristic acceptor that simulates every other heuristic acceptor.

Proof. It is a heuristic acceptor by Lemma 3.1 and Corollary 2.1. It simulates every other heuristic acceptor by Lemma 3.2, Proposition 2.2, and the fact that $U(x, 8d)$ simulates $U(x, d)$. □

4 Weakly automatizable heuristic proof systems

In this section we define heuristic proof systems and show that weakly automatizable heuristic proof systems are essentially equivalent to heuristic acceptors; hence, there is an optimal weakly automatizable heuristic proof system.

A randomized heuristic proof system has proofs of all "theorems" (these proofs are accepted with probability at least $\frac{1}{2}$) and has no proofs (even those accepted with probability $\frac{1}{8}$) of most "non-theorems" except for a $1/d$ fraction according to a sampler of "non-theorems". A formal definition follows.

Definition 4.1. A randomized Turing machine Π is a *heuristic proof system* for a 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 $\Pr_\Pi\{\Pi(x, w, d) = 1\} \geq \frac{1}{2}$. Every such string w is called a *correct* $\Pi^{(d)}$*-proof of* x.
3. (Soundness) $\Pr_{x \leftarrow D_n}\{\exists w : \Pr_\Pi\{\Pi(x, w, d) = 1\} \geq \frac{1}{8}\} < \frac{1}{d}$.

The main complexity characteristic of a heuristic proof system is the length of the shortest proof that is always accepted by this system. For a heuristic proof system Π, we denote $l_\Pi(x, d) = \min\{|w| \mid \Pr_\Pi\{\Pi(x, w, d) = 1\} \geq \frac{1}{2}\}$.

We now define the notion of a weakly automatizable proof system similarly to the classical case. Namely, a system is weakly automatizable if there is an algorithm that, given $x \in L$, finds efficiently a proof of x in *another* heuristic proof system and this proof is at most polynomially longer than the length of the shortest proof in Π.

Definition 4.2. A heuristic proof system Π is *weakly automatizable* if there is a randomized Turing machine A, a heuristic proof system $\hat{\Pi}$, and a polynomial p satisfying the following conditions:

1. For every $x \in L$ and every $d \in \mathbb{N}$,

$$\Pr_{w \leftarrow A(x,d)} \left\{ |w| \leq p(d \cdot |x| \cdot l_\Pi(x, d)) \wedge w \text{ is a correct } \hat{\Pi}^{(d)}\text{-proof of } x \right\} \geq \tfrac{1}{4}. \quad (2)$$

2. The running time of $A(x, d)$ is bounded by a polynomial in $|x|$, d, and the size of its own output.

A heuristic proof system Π is *automatizable* if it is weakly automatizable and $\hat{\Pi} = \Pi$.

Definition 4.3. We say that a heuristic proof system Π_1 *simulates* a heuristic proof system Π_2 if l_{Π_2} dominates l_{Π_1} on L.

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 case of acceptors.

Definition 4.4. A heuristic proof system Π is *polynomially bounded* if l_Π is polynomially bounded on L.

Proposition 4.1. If a heuristic proof system Π_1 simulates a heuristic proof system Π_2 and Π_2 is polynomially bounded, then Π_1 is also polynomially bounded.

Consider a weakly automatizable proof system $(\Pi, A, \hat{\Pi})$ for a distributional proving problem (D, L) with a recursively enumerable language L. Let us consider the following algorithm.

Algorithm 4.1 (Algorithm $U_\Pi(x, d)$)**.**

- Execute 1000 copies of $A(x, d)$ and a semidecision procedure for L in parallel.
- For each copy of $A(x, d)$,
  - if it stops with result w, then
    - execute $\hat{\Pi}(x, w, d)$ 60000 times;
    - accept if there were at least 10000 accepts of $\hat{\Pi}$ out of the 60000 runs.
- Accept if the semidecision procedure for L accepts.

Lemma 4.1. If $(\Pi, A, \hat{\Pi})$ is a heuristic weakly automatizable proof system for recursively enumerable language L, then U_Π is a heuristic acceptor for L and $l_\Pi(x, d)$ dominates $t_{U_\Pi}(x, d)$.

Proof. Soundness (condition 3 in Definition 2.2). Let $\Delta_n = \{x \in \overline{L} \mid \exists w : \Pr\{\hat{\Pi}(x, w, d) = 1\} \geq \frac{1}{8}\}$. By definition, $D_n(\Delta_n) < \frac{1}{d}$. For $x \in \overline{L} \setminus \Delta_n$ and specific w, Chernoff bounds imply that $\hat{\Pi}(x, w, d)$ accepts in at least $\frac{1}{6}$ fraction of the 60000 executions with exponentially small probability, which remains much smaller than $\frac{1}{8}$ even after multiplying by 1000.

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

Simulation. For $x \in L$, the probability that the event in (2) does not occur all 1000 times is negligible (at most $(\frac{3}{4})^{1000} < 10^{-120}$). Thus with high probability at least one of the parallel executions of $A(x, d)$ outputs a correct $\hat{\Pi}^{(d)}$-proof of size bounded by a polynomial in $l_\Pi(x, d)$, $|x|$, and d. For $x \in L$ and a correct $\hat{\Pi}^{(d)}$-proof w, the Chernoff–Hoeffding bound implies that $\hat{\Pi}(x, w, d)$ accepts in at least $\frac{1}{6}$ fraction of the 60000 executions with probability close to 1. Therefore, $t_{U_\Pi}(x, d)$ is bounded by a polynomial in $|x|$, d, and $l_\Pi(x, d)$. □

Lemma 4.2. Let C be a heuristic acceptor for (D, L). Then there is a weakly automatizable heuristic proof system Π_C for (D, L) such that t_C dominates l_{Π_C}.

Proof. The system Π_C will have unary words 1^T as proofs. Namely, $\Pi_C(x, 1^T, d)$ accepts if $C(x, d)$ accepts in at most T steps. The median time $t_C(x, d)$ written in unary is the shortest $\Pi_C^{(d)}$-proof of x. The correctness of Π_C follows from the correctness of C.

The weakly automatizing procedure for Π_C will output proofs that are accepted by this system with slightly smaller probability than the required $\frac{1}{2}$. Thus we construct another system $\hat{\Pi}$ that accepts these proofs. More precisely, $\hat{\Pi}(x, 1^T, d)$ executes ℓ parallel instances of $C(x, d)$ for at most T steps and stops as soon as at least $\frac{3}{16}\ell$ instances accept. For $x \in L$, the Chernoff–Hoeffding bound implies that for $T \geq t_C^{(1/4)}(x, d)$ and ℓ big enough, $\hat{\Pi}(x, 1^T, d)$ accepts with probability at least $\frac{1}{2}$.

For those inputs x for which $\Pr\{C(x, d) = 1\} < \frac{1}{8}$, the Chernoff–Hoeffding bound implies that $\Pr\{\hat{\Pi}(x, 1^T, d) = 1\}$ with exponentially small probability; in particular, this probability is less than $\frac{1}{8}$. The D-probability of other inputs is at most $\frac{1}{d}$, which implies the correctness of $\hat{\Pi}$.

The automatizing procedure executes $C(x, d)$ and, if it accepts, outputs 1^T where T is the number of steps made by C. With probability at least $\frac{1}{4}$ the resulting proof 1^T has $T \in [t_C^{(1/4)}(x, d); t_C^{(1/2)}(x, d)]$. Every such proof is a correct $\hat{\Pi}^{(d)}$-proof of length not exceeding $l_{\Pi_C}(x, d) = t_C^{(1/2)}(x, d)$. □

Theorem 4.1. For a recursively enumerable language L and a polynomial-time samplable distribution D, there is an optimal weakly automatizable heuristic proof system, i.e., one that simulates every other weakly automatizable heuristic proof system for (D, L).

Proof. By Lemma 4.1, every weakly automatizable heuristic proof system Π yields a heuristic acceptor A with $w_\Pi \succeq t_A$. Then, by Theorem 3.1, A is simulated by the universal heuristic acceptor U', i.e., $t_A \succeq t_{U'}$. By Lemma 4.2, the heuristic acceptor U' can be transformed into a weakly automatizable heuristic proof system $\Pi_{U'}$ with $t_{U'} \succeq w_{\Pi_{U'}}$. By transitivity, $\Pi_{U'}$ simulates Π. □

Remark 4.1. It is clear from the proof of Lemma 4.2 that one can omit the word "weakly" in the statement of Theorem 4.1 if the automatizing procedure is allowed to output not only proofs but also "almost proofs" accepted by the proof system with probability $\frac{1}{4}$.

5 Hard problems for heuristic acceptors

In this section, we show that the existence of one-way functions implies the existence of distributional proving problems that have no polynomially bounded heuristic acceptors. More precisely, the existence of such problems is equivalent to the existence of *infinitely-often* one-way functions, i.e., ones that are hard to invert for infinitely many input lengths. The logic of the equivalence is as follows: $\exists$ i.o. one-way functions $\Rightarrow$ $\exists$ i.o. pseudorandom generators $\Rightarrow$ $\exists$ intractable distributional proving problems $\Rightarrow$ $\exists$ average-case one-way functions $\Rightarrow$ $\exists$ i.o. one-way functions.

Definition 5.1. Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a length-preserving polynomial-time computable function. We call f *i.o. one-way* if for every polynomial-time randomized algorithm A and every polynomial p,

$$\forall n_0\ \exists n > n_0 \Pr_{x \leftarrow \mathbf{U}_n}\{A(x) \in f^{-1}(f(x))\} < \frac{1}{p(n)}.$$

Similarly to the classical case, the existence of i.o. one-way functions implies the existence of i.o. pseudorandom generators.

Definition 5.2. Let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function such that $|G(r)| = |r| + 1$ for every r. We call it an i.o. pseudorandom generator if for every polynomial-time randomized algorithm A and every polynomial p,

$$\forall n_0\ \exists n > n_0 \mid \Pr_{x \leftarrow \mathbf{U}_n}\{A(G(x)) = 1\} - \Pr_{x \leftarrow \mathbf{U}_{n+1}}\{A(x) = 1\}| < \frac{1}{p(n)}.$$

Theorem 5.1 (cf. [HILL99])**.** If there is an i.o. one-way function, then there is an i.o. pseudorandom generator.

Proof. The construction repeats that of [HILL99]. In the original construction the lengths of inputs where the one-way function is hard to invert are mapped to the lengths of inputs where the pseudorandom generator is hard to break. □

Assume that there exists an i.o. pseudorandom generator. We now show how to transform an i.o. pseudorandom generator into a distributional proving problem that has no polynomially bounded heuristic acceptors.

Theorem 5.2. If there is an i.o. pseudorandom generator G, then there is a distributional proving problem (D, L) with polynomial-time samplable D and $L \in$ **co-NP** such that there is no polynomially bounded heuristic acceptor for (D, L).

Proof. Define $D_{n+1}(x) = \Pr_{y \leftarrow \mathbf{U}_n}\{G(y) = x\}$ and $L = \bigcup_n (\{0,1\}^{n+1} \setminus G(\{0,1\}^n))$.

Suppose that there is a normalized heuristic acceptor A for (D, L) such that $t_A(x, d) \le q(|x|d)$ for a polynomial q. We then construct an algorithm $B(x)$ as follows: It executes $A(x, \frac{1}{10})$ for $q(10|x|)$ steps and outputs 1 iff A accepts; otherwise B outputs 0.

We now show that B breaks G. Indeed, $\Pr_{x \leftarrow \mathbf{U}_{n+1}}\{B(x) = 1\} \ge \frac{1}{2} \cdot \frac{|\{0,1\}^{n+1} \setminus G(\{0,1\}^n)|}{2^{n+1}} \ge \frac{1}{2}(1 - \frac{1}{2}) = \frac{1}{4}$. However, $\Pr_{x \leftarrow \mathbf{U}_n}\{B(G(x)) = 1\} < \frac{1}{10}$. Contradiction. □

Following [HI10], we define average-case one-way functions.

Definition 5.3. A length-preserving polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ is *average-case one-way* if for every poly$(|x|d)$-time randomized algorithm $A(x, d)$,

$$\forall n_0\ \exists n > n_0 \Pr_{x \leftarrow \mathbf{U}_n}[\Pr_A\{f(A(f(x), d)) \ne f(x)\} \ge \frac{1}{4}] \ge \frac{1}{d}.$$

Remark 5.1. By the same argument as in Propositions 2.2 and 2.3 one can show that the condition

$$\Pr_{x \leftarrow \mathbf{U}_n}\{\Pr_A\{f(A(f(x), d)) \ne f(x)\} \ge \frac{1}{4}\} \ge \frac{1}{d}$$

can be equivalently replaced by

$$\Pr_{x \leftarrow \mathbf{U}_n} \{f(A(f(x), d)) \neq f(x)\} \geq \frac{1}{d}.$$

Theorem 5.3 ([HI10])**.** The existence of average-case one-way functions implies the existence of i.o. one-way functions.

Theorem 5.4. Assume that (D, L) has no polynomially bounded heuristic acceptors. Then there exists an average-case one-way function.

Proof. Assume that D has a polynomial-time sampler g using $p(n)$ random bits for inputs of length n, i.e., $|g(x)| = n$ whenever $|x| = p(n)$. Define a function f as follows:

$$f(x) = \begin{cases} g(x)0^{p(n)-n} & \text{if } \exists n \ |x| = p(n), \\ x & \text{otherwise.} \end{cases}$$

We now show that f is average-case one-way. Let $B(x, d)$ be a $q(|x|d)$-time algorithm, where q is a polynomial, and for every n big enough, $\Pr_{x \leftarrow \mathbf{U}_{p(n)};\ B}\{f(B(f(x), d)) \neq f(x)\} < \frac{1}{d}$. Define an algorithm $A(x, d)$ as follows: compute $y = B(x0^{p(|x|)-|x|}, d)$; if $f(y) = x0^{p(|x|)-|x|}$, then output 0, otherwise 1.

The running time of $A(x, d)$ is poly$(|x|d)$. For $x \in L$, there is no preimage, i.e., $A(x, d) = 1$. For $x \notin L$, we have $\Pr_{x \leftarrow D_n;\ A}\{A(x, d) = 1\} = \Pr_{x \leftarrow \mathbf{U}_{p(n)};\ B}\{f(B(f(x), d)) \neq f(x)\} < \frac{1}{d}$, which gives a polynomially bounded normalized heuristic acceptor and thus a polynomially bounded heuristic acceptor for (D, L). □

Corollary 5.1. The existence of the following objects is equivalent:

- A distributional proving problem that has no polynomially bounded heuristic acceptors.
- An infinitely often one-way function.
- An average-case one-way function.
- An infinitely often pseudorandom generator.

Similarly to Levin's universal one-way function [Lev87], one can construct a universal distributional proving problem which is complete under reductions that preserve tractability for heuristic acceptors. Such reductions resemble reductions used in average-case complexity (see [BT06]).

Definition 5.4. A distributional proving problem (D_1, L_1) reduces to a distributional proving problem (D_2, L_2) if there is an injective polynomial-time computable function $f : \{0, 1\}^* \to \{0, 1\}^*$ mapping inputs of equal length to inputs of equal length such that

- $x \in L_1 \iff f(x) \in L_2$;
- there is a polynomial p such that for for every input x, $p(|x|)D_2(f(x)) \geq D_1(x)$.

Proposition 5.1. If (D, L) reduces to (D', L') and (D', L') has a polynomially bounded heuristic acceptor, then so does (D', L').

Proof. Assume that the reduction is given by a function f mapping words of length t into words of length $\varphi(t)$, and p is as in the definition above. Let $A(x, d)$ be a heuristic acceptor for (D', L'). Then $B(x, d) = A(f(x), p(|x|)d)$ is a heuristic acceptor for (D, L):

$$\Pr_{x \leftarrow D_n}\{\Pr_A\{A(f(x), p(|x|)d) = 1\} \geq \tfrac{1}{8}\} = \sum_{\substack{x \in \operatorname{supp} D_n: \\ \Pr\{A(f(x),p(n)d)=1\} \geq \frac{1}{8}}} D(x)$$

$$\leq \sum_{\substack{x \in \operatorname{supp} D_n: \\ \Pr\{A(f(x),p(n)d)=1\} \geq \frac{1}{8}}} p(n) D'(f(x)) \leq \sum_{\substack{y \in \operatorname{supp} D'_{\varphi(n)}: \\ \Pr\{A(y,p(n)d)=1\} \geq \frac{1}{8}}} p(n) D'(y)$$

$$= p(n) \cdot \Pr_{y \leftarrow D'_{\varphi(n)}}\{\Pr\{A(y, p(n)d) = 1\} \geq \tfrac{1}{8}\} < p(n)/(dp(n)) = 1/d.$$

□

Lemma 5.1. There is a constant C such that every problem (D, L) reduces to some problem (D', L') where D' has a sampler running in time at most Cn^2.

Proof. We use padding. Assume that a sampler g for D runs in at most cn^c steps and uses at least n random bits. The new sampler h, asked to produce a sample of length n, pads g's sample by outputting $h(r) = 0^{cn^c} 1 g(r)$. Let $L' = \{0^{c|x|^c} 1x \mid x \in L\}$. The reduction is given by $f(y) = 0^{c|y|^c} 1y$. □

We now construct a universal distributional proving problem (R, X). The language X contains only inputs of even lengths. The distribution R is uniform on odd lengths and defined on length $2n$ as follows: with probability $1/2^i$ (where $1 \leq i \leq n-1$), its sampler $g(r)$ outputs the concatenation $s_i A_i(r)$, where s_i is the word of length n that has zeroes in all positions except i, where it has 1, and A_i is the algorithm with Goedel number i equipped with a quadratic-time alarm clock as in Lemma 5.1; with probability $1/2^{n-1}$ it outputs $s_n A_n(r)$. Let $X = \{0,1\}^* \setminus \operatorname{supp} R$.

Theorem 5.5. Every distributional proving problem (D, L) reduces to (R, X).

Proof. By Lemma 5.1, the problem (D, L) reduces to a quadratic-time samplable problem (D', L'). Assume that the sampler for D' has Goedel number k; then (D', L') reduces to (R, X) by $f(x) = s_k x$, where s_k is as in the definition of R (for $|x| < k$, f computes the answer itself and maps x to an appropriate fixed string, which takes a constant time). The domination condition $2^k R(x) \geq D'(x)$ is satisfied, since k is a constant. □

6 Further research

- For a weakly automatizable heuristic proof system equivalent to a heuristic acceptor, show that its automatizing procedure can output a correct proof in the same proof system. (That is, the system is automatizable.)
- Devise an optimal heuristic proof system.

- Suggest a nice conjecture implying the existence of distributional proving problems that have no polynomially bounded heuristic proof systems.
- Consider a version of the notion of a heuristic proof system related to trapdoor functions.

Acknowledgements

The authors are grateful to Dima Antipov and Dima Grigoriev for helpful discussions and to anonymous referees for comments that improved the quality of the paper.

References

[ABSRW00] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. Technical Report 00-023, Electronic Colloquium on Computational Complexity, 2000. Extended abstract appears in Proceedings of FOCS-2000.

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

[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.

[HI10] E. A. Hirsch and D. M. Itsykson. An infinitely-often one-way function based on an average-case assumption. *St. Petersburg Math. J.*, 21(3):459–468, 2010.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[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.

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

[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.

[Kra01a] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1–3):123–140, 2001.

[Kra01b] Jan Krajíček. Tautologies from pseudorandom generators. *Bulletin of Symbolic Logic*, 7(2):197–212, 2001.

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

[Lev87] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7, 1987.

[McD98] C. McDiarmid. *Concentration*, volume 16 of *Algorithms and Combinatorics*, pages 195–248. Springer-Verlag, 1998.

[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.

[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.

[Sad99] Zenon Sadowski. On an optimal deterministic algorithm for SAT. In *Proceedings of CSL'98*, volume 1584 of *Lecture Notes in Computer Science*, pages 179–187. Springer, 1999.