Optimal acceptors and optimal proof systems

Edward A. Hirsch*

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

Abstract. Unless we resolve the **P** vs **NP** question, we are unable to say whether there is an algorithm (*acceptor*) that accepts Boolean tautologies in polynomial time and does not accept non-tautologies (with no time restriction). Unless we resolve the **co-NP** vs **NP** question, we are unable to say whether there is a proof system that has a polynomial-size proof for every tautology.

In such a situation, it is typical for complexity theorists to search for "universal" objects; here, it could be the "fastest" acceptor (called *optimal acceptor*) and a proof system that has the "shortest" proof (called *optimal proof system*) for every tautology. Neither of these objects is known to the date.

In this survey we review the connections between these questions and generalizations of acceptors and proof systems that lead or may lead to universal objects.

1 Introduction and basic definitions

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 instances of 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.

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 a certain language L; if $L = \mathtt{TAUT}$ 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 the formula is a tautology: it must accept at

* Partially supported by RFBR grant 08-01-00640, the president of Russia grant "Leading Scientific Schools" NSh-5282.2010.1, and by Federal Target Programme "Scientific and scientific-pedagogical personnel of the innovative Russia" 2009–2013.

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 proof for every tautology in Π_s is at most polynomially longer than its shortest proof 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) proof system is a major open question for many languages including TAUT. Optimality would imply p-optimality for any system and any language if and only if the natural proof system for SAT (satisfying assignments) is p-optimal; the existence of optimal system would imply the existence of p-optimal system if there is some p-optimal system for SAT [BKM09a]. If an optimal system for TAUT 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]. The existence of a p-optimal system for quantified Boolean formulas would imply a complete language in $\mathbf{NP} \cap \mathbf{co\text{-}NP}$ [Sad97]. (See [BS09] regarding the situation for other languages and complexity classes.) Unfortunately, no concise widely believed structural assumptions (like $\mathbf{NP} \neq \mathbf{co\text{-}NP}$) are known to imply the (non-)existence of (p-)optimal proof systems. Krajíček and Pudlák [KP89] showed that the existence is implied by $\mathbf{NE} = \mathbf{co\text{-}NE}$ (resp., $\mathbf{E} = \mathbf{NE}$) for optimal (resp., p-optimal) propositional proof systems, and Köbler, Messner, and Torán [KMT03] weakened these conjectures to doubly exponential time, but these conjectures are not widely believed.

An *acceptor* for a language L is a semidecision procedure, i.e., an algorithm that answers 1 for $x \in L$ and does not stop for $x \notin L$. An acceptor O is *optimal* if for any other (correct) acceptor A, for every $x \in L$, the acceptor O stops on x in time bounded by a polynomial in $|x|$ and the time taken by $A(x)$. (In [KP89] optimal acceptors are called p-optimal algorithms; the term "acceptor" was later introduced by Messner in his PhD thesis). Krajíček and Pudlák [KP89] showed that for TAUT the existence of a p-optimal system is equivalent to the existence of an optimal acceptor. 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. We review these results in **Section 3**. Monroe [Mon09] recently formulated a conjecture implying that such an algorithm does not exist[1]. Note that Levin [Lev73] showed that an optimal algorithm does exist for finding witnesses for SAT (equivalently, for non-tautologies): just run all algorithms in parallel and stop as soon as one of them returns a satisfying assignment. However, this procedure gives an optimal acceptor neither for TAUT nor for SAT, because (1) on tautologies, it simply does not stop; (2)

[1] More precisely, if an optimal acceptor for TAUT exists, then there is an acceptor for $\overline{\mathtt{BH}}$ (where $\mathtt{BH} = \{(M, x, 1^t) \mid \text{nondeterministic Turing machine } M \text{ accepts } x \text{ in } t \text{ steps}\}$) that works in time polynomial in t for every particular (M, x) such that M does not stop on x.

Levin's algorithm enumerates *search* algorithms, and for an acceptor we need *decision* algorithms, which may be faster for some inputs; the search-to-decision reduction adds a lot to the running time by running the decision algorithm for *shorter formulas as well*, which may be surprisingly much larger than for the original input.

A proof system Π is *automatizable* if it has an automatization procedure that works in time bounded by a polynomial in the output length. This procedure A, given a "theorem", outputs its (correct) proof for Π of length polynomially bounded by the length of the shortest proof for Π. It is easy to see that such a proof system can be easily turned into an acceptor with running time polynomially related to the proof size and the input size. Vice versa, an acceptor can be converted into an automatizable proof system, where the proof is just the number of steps (written in unary) that the acceptor makes before accepting its input. Thus, in the classical case there is no difference between acceptors and automatizable proof systems.

Extensions that give optimality. An obvious obstacle to constructing an optimal proof system or an optimal acceptor by enumeration is that no efficient procedure is known for enumerating the set of all complete and sound proof systems (resp., acceptors). Recently, similar obstacles were overcome in other settings by considering either computations with *non-uniform advice* (see [FS06] for a survey) or *heuristic* algorithms [FS04,Per07,Its09]. In particular, (p-)optimal proof systems (and acceptors) with advice do exist [CK07], and we review this fact in **Section 4**.

As to heuristic computations, the situation is more complex. Recently, it was proved [HI10] that an optimal randomized heuristic acceptor does exist. However, in the heuristic case we lack the equivalence between optimal proof systems and optimal acceptors, and even the equivalence between acceptors and automatizable proof systems is not straightforward. We review the heuristic case (including more recent directions) in **Section 5**.

Another possibility to obtain optimal proof systems is to generalize the notion of simulation. Recently, Pitassi and Santhanam [PS10] suggested a notion of "effective simulation" and constructed a proof system for quantified Boolean formulas that effectively simulates all other proof systems for `QBF`. We do not review this result here.

We conclude the paper by listing open questions in **Section 6**.
We now continue to **Section 2** listing trivial facts that we will use in what follows.

2 Trivia

Enumeration. Almost all constructions used in this survey employ the enumeration of all Turing machines of certain kind. Some remarks regarding this follow.

First of all, recall that deterministic Turing machines (either decision machines that say yes/no, or transducers that compute arbitrary functions) can

be efficiently enumerated by their Gödel numbers and simulated with only a polynomial overhead in time.

For a recursively enumerable language, one can enumerate acceptors only by running the semidecision procedure in parallel.

Also one can require, if necessary, that the construction of each machine (we will need it for proof systems) includes an "alarm clock" that interrupts the machine after a certain number of steps.

Each proof system Π is p-simulated by another proof system Π' where $\Pi'(x, w)$ runs in time, say, $100(|x| + |w|)^2$: just pad the proofs appropriately; the simulation omits the padding.

These facts allow us to limit ourselves to enumerating proof systems with quadratic alarm clock.

Frequently, we write that we execute "in parallel" a large (sometimes infinite) number of computations. In reality this is achieved, of course, sequentially by alternating consecutive steps of these computations (for example, simulating the step k of the i-th machine at step $(1 + 2k) \cdot 2^i$, as in Levin's optimal algorithm).

Acceptors and proof systems for subsets. For recursively enumerable L, for every acceptor A' for $L' \subseteq L$ there is an acceptor A for L that is almost as efficient on L' as A'. Similarly, for every proof system Π' for $L' \subseteq L$ there is a proof system Π for L that has the same proofs on L' as Π'.

Proofs vs candidate proofs. In what follows we call w a Π-proof of x if $\Pi(x, w) = 1$. Sometimes we write "u is a candidate Π-proof of x" to emphasize that u is intended for checking with Π (while it is not yet known whether $\Pi(x, u) = 1$).

3 Optimal acceptors exist iff p-optimal proof systems exist

3.1 Krajíček-Pudlák's proof

In this section we give the proof by Krajíček and Pudlák [KP89]. The original exposition demonstrates the equivalence of many statements, while we need only two and show only the required implications.

Theorem 1 ([KP89]). *Optimal acceptors for* TAUT *exist iff p-optimal proof systems for* TAUT *exist. Moreover, the sufficiency (⇐) holds for any language, not just* TAUT.

Proof. $\boxed{\Rightarrow}$. A candidate proof of x for our p-optimal proof system Π_* contains a description of a proof system Π (i.e., a deterministic Turing machine given by its Gödel number and equipped with a quadratic alarm clock) and a candidate Π-proof π. To verify the proof, $\Pi_*(x, (\Pi, \pi))$ simply simulates $\Pi(x, \pi)$ ensuring that Π accepts π and then verifies the correctness of Π by querying the optimal acceptor A_* for the statement

$$\forall y \in \{0,1\}^{|x|}\ \forall \psi \in \{0,1\}^{|\pi|}\ \forall z \in \{0,1\}^{|x|}\ (\Pi(y, \psi) = 0 \vee y[z] = 0)$$

written as a Boolean formula (here $y[z]$ denotes the result of substituting the consecutive bits of z for the consecutive variables of the Boolean formula y).

Since for every Π there is an algorithm that, given $|x|$ and $|\pi|$, writes such statement in time polynomial in $|x| + |\pi|$, A_* must stop in (specific) polynomial time for specific (correct) Π, which proves that Π_* p-simulates Π.

$\boxed{\Leftarrow}$. The optimal acceptor A_* just applies in parallel all deterministic transducers hoping one of them outputs a Π_*-proof and lets Π_* verify the result. Once Π_* returns 1, the acceptor A_* stops.

Since every acceptor A is a particular case of a proof system, there is a polynomial-time algorithm that, given tautology x, outputs Π_*-proofs in time polynomial in $|x|$ and time spent by A on x. □

Remark 1. Sadowski [Sad07] explains that there exist an optimal automatizable proof system for TAUT iff there is a (deterministic) acceptor that is optimal for non-deterministic acceptors as well.

3.2 Messner's proof

Messner [Mes99] generalized the result of Krajíček and Pudlák to a wider class of languages. His proof is also interesting even in the TAUT case, because it replaces the statement about the correctness of a proof system on all inputs of certain size by the statement about the correctness of a single proof of a single input.

Definition 1 ([BH77]). *A language L is* paddable *if there is an injective non-length-decreasing polynomial-time padding function* $\mathrm{pad}_L\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *that is polynomial-time invertible on its image and such that for every x and w,*

$$x \in L \iff \mathrm{pad}_L(x, w) \in L.$$

Theorem 2 ([Mes99]). *For every paddable r.e. language L, optimal acceptors for L exist iff p-optimal proof systems for L exist.*

Proof. $\boxed{\Rightarrow}$. Similarly to Krajíček-Pudlák's proof, a candidate proof for our p-optimal proof system Π_* contains a description of a proof system Π (with a quadratic alarm clock) and a candidate Π-proof π, and $\Pi_*(x, (\Pi, \pi))$ starts the verification by simulating $\Pi(x, \pi)$. What makes a difference is how Π_* verifies the correctness of Π.

One could simulate the optimal acceptor A_* on x restricting its running time to a certain polynomial of $|x| + |\pi|$. However, there is no warranty that this amount of time is enough for A_*. Therefore, we run it on a different input where A_* is guaranteed to run in polynomial time and certifies the correctness of the proof π. Namely, we run it on $\mathrm{pad}_L(x, \pi)$. By the definition of pad_L, the result is 1 iff $x \in L$. For a correct proof π, this result is output in a polynomial time because for a correct system Π, the set $\{\mathrm{pad}_L(x, \pi) \mid x \in L,\ \Pi(x, \pi) = 1\} \subseteq L$ can be accepted in a polynomial time (the polynomial is the sum of the time spent by pad_L^{-1} and by Π), and A_* is an optimal acceptor.

$\boxed{\Leftarrow}$. See Theorem 1. □

4 Non-uniform advice gives optimal proof systems

Cook and Krajíček [CK07] show that allowing one bit of non-uniform advice yields a p-optimal proof system (against simulations with advice). A similar result for acceptors is not known.

Definition 2. *A proof system with $t(n)$ bits of advice is a polynomial-time algorithm $\Pi(x, w, \alpha)$ and a sequence $(\alpha_n)_{n\in\mathbb{N}}$, where $\alpha_n \in \{0,1\}^{t(n)}$, such that for all x,*

$$x \in L \iff \exists w\ \Pi(x, w, \alpha_{|x|+|w|}) = 1.$$

Theorem 3 ([CK07], see also [BKM09b]). *For every language L, there is a proof system with 1 bit of advice that simulates every proof system with $k(n) = O(\log n)$ bits of advice. Moreover, the simulation can be computed in polynomial time with $k(n)$ bits of advice.*

Proof. A candidate proof for the constructed proof system Π_* contains a description of a proof system Π (a deterministic Turing machine given by its Gödel number and equipped with a quadratic alarm clock) *written in unary* as 1^Π, a candidate Π-proof π, and an advice string $\alpha \in \{0,1\}^{k(n)}$ written in unary as a string 1^α of length $\leq 2^{k(n)}$. Then $\Pi_*(x, (1^\Pi, \pi, 1^\alpha))$ starts the verification by simulating $\Pi(x, \pi, \alpha)$. To verify the correctness of Π, the system Π_* simply queries its advice bit, which is supposed to say whether Π with advice string α is correct on all candidate theorems of size $|x|$ and proofs of size $|\pi|$. To ensure that this is the same bit for all couples $(x, (1^\Pi, \pi, 1^\alpha))$ of the same size, we must choose pairing function such that for all strings $a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4$, for each i, $|a_i| \neq |b_i| \Rightarrow |(a_1, (a_2, a_3, a_4))| \neq |(b_1, (b_2, b_3, b_4))|$.

The simulation can be computed trivially. □

Remark 2. One may suppose that, similarly to the classical case, the existence of optimal proof systems with advice implies the existence of disjoint **NP** pairs with advice. This is indeed the case; however, in order to keep the closedness under reductions (with advice) the advice given must have length $O(1)$ and not a specific constant number of bits. For exactly one bit of advice one gets only an **NP** pair that is *hard* for disjoint **NP** without advice under many-one reductions without advice [BS09].

5 Heuristic case: optimal acceptors exist, hope for proof systems

Hirsch and Itsykson [HI10] introduced heuristic[2] acceptors (called heuristic automatizers in [HI10]) and heuristic proof systems. Heuristic algorithms (see, e.g.,

[2] All heuristic computations that we consider are randomized. Therefore, we omit the word "randomized" from the terms that we introduce and mention it explicitly only in the definitions. However, it is possible to consider deterministic heuristic acceptors and proof systems as well.

[BT06]) are algorithms that make errors for a small amount of inputs. Similarly, heuristic proof systems claim a small amount of wrong "theorems".

Since we are interested in the behaviour only on the positive instances ("theorems"), our definition of a distributional problem is different from the usual definition used for heuristic algorithms. 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 proof system is small.

Definition 3. *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.

5.1 Heuristic acceptors

Definition 4. *A* heuristic acceptor *for 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 outputs 1 (denoted $A(\ldots) = 1$) or does not halt at all;*
2. *For every $x \in L$ and $d \in \mathbb{N}$, $A(x, d) = 1$.*
3. *For every $n, d \in \mathbb{N}$,*

$$\Pr_{r \leftarrow D_n} \left\{ \Pr_A \{A(r, d) = 1\} > \frac{1}{8} \right\} < \frac{1}{d}.$$

(In fact, the specific constant is not important.)

Remark 3. Similarly to the classical case, 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.

Given the definition of heuristic acceptor, we now adapt the classical notions of simulation and optimality to the heuristic case and give related basic facts. In what follows, all acceptors are for the same problem (D, L).

Definition 5. *The* time *spent by heuristic acceptor A on input (x, d) is defined as the median time*

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

Definition 6. *Heuristic acceptor S simulates heuristic acceptor W if there are polynomials p and q such that for every $x \in L$ and $d \in \mathbb{N}$,*

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

An optimal *heuristic acceptor is one that simulates every heuristic acceptor.*

Definition 7. *Heuristic acceptor 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 1.

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

Remark 4 (I.Monakhov). If one-way functions exist, then there is a polynomial-time samplable distribution D such that $(D, \texttt{TAUT})$ has no polynomially bounded heuristic acceptor.

Theorem 4 ([HI10]). *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 heuristic acceptor for (D, L).*

Proof (sketch). The construction consists of three procedures.

The first one, Test, estimates the probability of error of a candidate acceptor A on a given input by repeating A and counting its errors, it accepts if the number of errors is above certain threshold.

The second one, Certify, makes sure that the outermost probability in the correctness condition of a candidate acceptor A is small enough by repeating A at randomly sampled inputs of prescribed size, and counting errors reported by Test.

Finally, the optimal acceptor U, given x and d, runs the following processes for $i \in \{1, \ldots, l(n)\}$, where $l(n)$ is any slowly growing function, in parallel:

1. For certain d', run $A_i(x, d')$, the algorithm with Gödel number i satisfying conditions 1 and 2 of Def. 4, and compute the number of steps T_i made by it before it stops.
2. If Certify accepts A_i executed on inputs of size $|x|$ for at most T_i steps, then output 1 and stop U (all processes).

If none of the processes has stopped, U goes into an infinite loop. □

5.2 Heuristic proof systems

Definition 8. *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* correct $\Pi^{(d)}$-proof of x.
3. *(Soundness)* $\Pr_{x \leftarrow D_n}\{\exists w : \Pr\{\Pi(x,w,d)=1\} > \frac{1}{8}\} < \frac{1}{d}$.

Unfortunately, we cannot prove the equivalence between acceptors and proof systems in the heuristic case. Therefore, we focus our attention on automatizable heuristic proof systems. Surprisingly, even the equivalence between acceptors and automatizable proof systems is not straightforward in this case.

In [HI10], a heuristic automatizable proof system is defined straightforwardly, and it is shown that it necessarily defines an acceptor taking time at most polynomially larger than the length of the shortest proof in the initial system. This shows that heuristic acceptors form a more general notion than automatizable heuristic proof systems. Neither the converse nor the existence of an optimal automatizable heuristic proof system is shown in [HI10].

However, for a relaxed definition presented below, the equivalence does take place. (The proof of this statement is not published yet [HIS10], so you should take it with a bunch of salt for now.) In particular, there exists a p-optimal system in the class of heuristic automatizable proof systems.

Definition 9. *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 $d \in \mathbb{N}$, there is a polynomial p such that*

$$\Pr_{w \leftarrow A(x,d)}\left\{|w| \leq p(d \cdot |x| \cdot |w_*|) \wedge \Pr\{\Pi(x,w,d)=1\} \geq \tfrac{1}{4}\right\} \geq \tfrac{1}{4}, \qquad (1)$$

   where w_* is the shortest correct $\Pi^{(d)}$-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.*

Remark 5. 1. This definition is different from one in [HI10].

2. We do not require the algorithm A to generate correct proofs. It suffices to generate "quasi-correct" (such that $\Pr\{\Pi(x,w,d)=1\} \geq \frac{1}{4}$) with probability $\frac{1}{4}$. At present, we are unable to construct an optimal system or to show the equivalence to heuristic acceptors if $\frac{1}{4}$ is strengthened to $\frac{1}{2}$ as in [HI10].

Definition 10. *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 correct $\Pi_1^{(d)}$-proof of x has size at most*

$$p(d \cdot |x| \cdot \max_{d' \leq q(d \cdot |x|)}\{\textit{the size of the shortest correct } \Pi_2^{(d')}\textit{-proof of } x\}). \qquad (2)$$

We say that heuristic proof system Π_1 p-simulates[3] *heuristic proof system Π_2 if Π_1 simulates Π_2 and there is a polynomial-time (deterministic) algorithm that*

[3] The definition we give here may be too restrictive: it requires a *deterministic* algorithm that is given a proof for *specific* $d' = q(d \cdot |x|)$. It works for our purposes, but can be relaxed.

converts a $\Pi_2^{q(d\cdot|x|)}$-proof into a $\Pi_1^{(d)}$-proof of size at most (2) such that the probability to accept a new proof is no smaller than the probability to accept the original one.

Definition 11. *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 correct $\Pi^{(d)}$-proof of x is bounded by $p(d \cdot |x|)$.*

Proposition 2. *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 heuristic acceptors 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 $\Pi(x, w, d)$ 60000 times;
   - if there were at least 10000 accepts of Π (out of 60000), 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 3. *If (Π, A) is a (correct) heuristic automatizable proof system for recursively enumerable language L, then A_Π is a (correct) heuristic acceptor for $x \in L$ and $t_{A_\Pi}(x, d)$ is bounded by polynomial in the size of the shortest correct $\Pi^{(d)}$-proof of x.*

The proof is a routine application of Chernoff bounds, and we skip it. The converse statement is also true. The construction of automatizable heuristic proof system (Π, A) made from heuristic acceptor B is as follows.

$A(x, d)$:

1. Run $B(x, d)$.
2. If $B(x, d) = 1$, output 1^T, where T is the total number of steps made by B.

$\Pi(x, w, d)$:

1. Run $B(x, d)$.
2. If B makes less than $|w|$ steps and outputs a 1, accept. Otherwise, reject.

Theorem 5. *(Π, A) is an automatizable heuristic proof system. For every $x \in L$, the length of the shortest Π-proof is upper bounded by the (median) running time of B.*

Corollary 1 (optimal automatizable heuristic proof system). *There is an automatizable heuristic proof system that p-simulates every automatizable heuristic proof system.*

6 Open questions

Classical systems. It is still unknown whether (p-)optimal propositional proof systems exist. No concise widely believed structural assumption (such as $\mathbf{NP} \neq \mathbf{co\text{-}NP}$) is known to imply their nonexistence.

Acceptors with advice. Construct an optimal acceptor with short non-uniform advice either by extending the equivalence between optimal acceptors and p-optimal proof systems or directly. The main obstacle here is the difference between proof systems with "input advice" (as defined above) and proof systems with "output advice" (where an advice string corresponds to the length of input x, not proof w). However, [BKM09b] shows that for propositional proof systems with logarithmic advice, these cases are equivalent w.r.t. the polynomial boundedness.

Heuristic systems. It is challenging to extend the equivalence between optimal acceptors and p-optimal proof systems to the heuristic case, as it would give an optimal heuristic system.

It would be interesting to strengthen the automatizability condition for heuristic systems by requiring to output real proofs (i.e., replace the innermost probability $\frac{1}{4}$ by $\frac{1}{2}$) without losing the equivalence to heuristic acceptors.

It would also be interesting to suggest a notion of a heuristic disjoint **NP** pair and show the existence of a complete pair.

Acknowledgements

The author is very grateful to Olaf Beyersdorff, Dmitry Itsykson, Hunter Monroe, Sergey Nikolenko, Natalia Tsilevich, Zenon Sadowski, and Alexander Smal for valuable comments, and to all participants of the proof complexity seminar in PDMI in September–December 2009 for numerous discussions on the topic.

References

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[BKM09a] Olaf Beyersdorff, Johannes Köbler, and Jochen Messner. Nondeterministic functions and the existence of optimal proof systems. *Theor. Comput. Sci.*, 410(38-40):3839–3855, 2009.

[BKM09b] Olaf Beyersdorff, Johannes Köbler, and Sebastian Müller. Nondeterministic instance complexity and proof systems with advice. In *Proceedings of LATA'09*, volume 5457 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2009.

[BS09] Olaf Beyersdorff and Zenon Sadowski. Characterizing the existence of optimal proof systems and complete sets for promise classes. In *Proceedings of CSR-2009*, volume 5675 of *Lecture Notes in Computer Science*, pages 47–58. Springer, 2009.

[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, 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.

[HI10] Edward A. Hirsch and Dmitry Itsykson. On optimal heuristic randomized semidecision procedures, with application to proof complexity. In *Proceedings of STACS 2010*, pages 453–464, 2010.

[HIS10] Edward A. Hirsch, Dmitry Itsykson, and Alexander Smal. Optimal heuristic randomized acceptors and automatizable proof systems. Unpublished manuscript, March 2010.

[Its09] Dmitry M. Itsykson. Structural complexity of AvgBPP. In *Proceedings of CSR-2009*, volume 5675 of *Lecture Notes in Computer Science*, pages 155–166, 2009.

[KMT03] Johannes Köbler, Jochen Messner, and Jacobo Torán. Optimal proof systems imply complete sets for promise classes. *Inf. Comput.*, 184(1):71–92, 2003.

[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. In Russian. English translation in: B.A.Trakhtenbrot. A Survey of Russian Approaches to Perebor (Brute-force Search) Algorithms. *Annals of the History of Computing* 6(4):384-400, 1984.

[Mes99] Jochen Messner. On optimal algorithms and optimal proof systems. In *Proceedings of STACS-99*, 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.

[PS10] Toniann Pitassi and Rahul Santhanam. Effectively polynomial simulations. In *Proceedings of ICS-2010*, 2010.

[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.

[Sad97] Zenon Sadowski. On an optimal quantified propositional proof system and a complete language for NP∩co-NP. In *Proceedings of FCS'97*, volume 1279 of *Lecture Notes in Computer Science*, pages 423–428. Springer, 1997.

[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.

[Sad07] Zenon Sadowski. Optimal proof systems, optimal acceptors and recursive representability. *Fundamenta Informaticae*, pages 169–185, 2007.