Записки научных
семинаров ПОМИ
Том 399, 2012 г.

E. A. Hirsch, D. M. Itsykson, V. O. Nikolaenko, A. V. Smal

OPTIMAL HEURISTIC ALGORITHMS FOR THE IMAGE OF AN INJECTIVE FUNCTION

Abstract. The existence of optimal algorithms is not known for any *decision* problem in $\mathbf{NP}\setminus\mathbf{P}$. We consider the problem of testing the membership in the image of an injective function. We construct optimal *heuristic* algorithms for this problem in both randomized and deterministic settings (a heuristic algorithm can err on a small fraction $\frac{1}{d}$ of the inputs; the parameter d is given to it as an additional input). Thus for this problem we improve an earlier construction of an optimal *acceptor* (that is optimal on the negative instances only) and also give a deterministic version.

§1. Introduction

1.1. Optimal algorithms. When we face a computational problem that is not known to be solved in a reasonable (say, polynomial) amount of time, we are still interested to solve it as fast as possible. The existence of an *optimal* algorithm that for *every possible input* returns its answer at least as fast (up to a polynomial) as any other algorithm for the same problem does, is an important structural feature of the problem and the model of computation (deterministic algorithms, bounded-error randomized algorithms, etc.).

While Levin's optimal algorithm for **NP** search problems is known for decades [8], it does not give an optimal algorithm for any decision problem, because, while for **NP**-complete problems the *worst-case* complexity of search and decision are polynomially related, a decision algorithm still can be exponentially faster for some inputs. Also Levin's algorithm does not stop at all on the negative instances. For many interesting languages including the language of Boolean tautologies TAUT, the existence of an

Key words and phrases: optimal algorithm, heuristic algorithm.

Supported in part by Federal Target Programme "Scientific and scientific-pedagogical personnel of the innovative Russia" 2009-2013, by the grants NSh-3229.2012.1 and MK-4108.2012.1 from the President of RF, by the Programme of Fundamental Research of RAS, and by RFBR grant 11-01-12135-ofi-m-2011. The second author is also supported by Rokhlin Fellowship.

algorithm that is optimal on the positive instances only (such algorithm is called an *optimal acceptor*) is equivalent to the existence of a p-optimal proof system (that is, a proof system that has the shortest possible proofs, and these proofs can be constructed by a polynomial-time algorithm given proofs in any other proof system) [7,10,12] (see [6] for survey). Chen, Flüm, and Müller [2] recently proved that the existence of optimal acceptors for all **co-NP**-complete languages is equivalent. Monroe [11] recently showed that proving the nonexistence of optimal acceptors for a **co-NP**-complete language is equivalent to proving that for every Turing machine M accepting a complement to the bounded halting problem there is a single "counterexample", i.e., a single machine N_M and its input x_M that force M to work on $(N_M, x_M, 1^t)$ longer than any polynomial in t.

1.2. Optimal heuristic randomized acceptors. An obvious obstacle to constructing an optimal algorithm by enumeration is that no efficient procedure is known for enumerating the set of all correct algorithms for, say, `TAUT` or `SAT`. A possible workaround is to check the correctness for a particular input; however, even for `SAT`, a search-to-decision reduction maps the input instance to a *different* instance and thus potentially increases the complexity.

The correctness can be, however, checked in the heuristic setting. A heuristic algorithm for a language L and probability distribution D on the inputs is allowed to make errors for some inputs; the probability of error according to D must be kept below $\frac{1}{d}$, where d is an integer parameter given to the algorithm. In [5] an optimal heuristic randomized acceptor for every r.e. language L and every polynomial-time samplable D *concentrated on* $\overline{L}$ is constructed. In other words, this is an algorithm that accepts (with bounded probability of error) every $x \in L$ in the fastest possible way, and accepts $x \notin L$ for inputs of total D-probability at most $\frac{1}{d}$.

1.3. Our results: derandomization and optimal heuristic algorithms. In this paper we consider the decision problem for the image of an injective function (under the uniform distribution) that maps n-bit strings to $(n+1)$-bit strings. Its study is motivated, for example, by the fact that a particular case of this problem is the problem of recognizing the image of an injective pseudorandom generator, which has no polynomial-time heuristic randomized algorithm [5, Theorem 5.2]. It is known that injective pseudorandom generators exist if one-way permutations exist [3].

For this problem, we extend the previous results in two directions. First, we devise an optimal algorithm, while [5] gave a construction of an optimal acceptor. Note that optimal algorithms halt on all inputs and optimality is defined for all inputs, not just the positive ones. In [5], the correctness test was performed by repeated sampling inputs in $\overline{L}$ and running a candidate acceptor on them. In our case $\overline{L}$ is the image of an injective function and we can still sample it. However, we still do not have a samplable distribution on L, i.e., on the complement to the image. The check is then done by testing the algorithm on a random input from $\{0,1\}^n$ and computing its overall probability of acceptance.

Our second result is a derandomization of this construction, namely, a deterministic algorithm that is optimal on the average. To do this, we use an expander-based construction of Goldreich and Wigderson [4] of small families of functions with good mixing properties, and also use the input as a source of pseudorandomness. It also derandomizes the construction of [5] of optimal acceptors if we consider it for the same class of problems (i.e., recognizing the complement of the image of an injective function).

A byproduct of the derandomization is the existence of an optimal automatizable proof system for the complement of the image. For our problem, this extends [5, Theorem 4.1], where only an optimal *weakly automatizable* randomized heuristic proof system is constructed, i.e., a proof system where the automatization procedure outputs a proof in a stronger system. (The necessary definitions and the corollary are given in Sec. 6.)

1.4. Organization of the paper. In Sec. 2, we give the necessary definitions. Then, in Sec. 3, we give a general construction of an optimal algorithm that suits both the deterministic and randomized cases but misses an important part: the procedure for estimating the frequency of a particular answer of an algorithm on a particular distribution of the inputs. In Sec. 4, we give a (rather simple) randomized testing procedure, and in Sec. 5, we give a (somewhat more complicated) deterministic one. Finally, we present directions for further research in Sec. 7.

§2. Definitions

2.1. Basic notation. An *ensemble* of probability distributions is a sequence of probability distributions $\{D_n\}_{n\in\mathbb{N}}$, where D_n is concentrated on $\{0,1\}^n$. We will denote such an ensemble by a single letter D and abuse the language by calling D a *distribution*.

Let U denote the ensemble $\{U_n\}_{n\in\mathbb{N}}$, where U_n is uniformly distributed on $\{0,1\}^n$. For every language $L \subseteq \{0,1\}^*$, we denote the uniform distribution on $L \cap \{0,1\}^n$ by $U_n(L)$; then $U(L) = \{U_n(L)\}_{n\in\mathbb{N}}$.

A *distributional problem* is a pair (L, D) consisting of a language $L \subseteq \{0,1\}^*$ and a distribution D.

We use subscripts to denote the probability space; for example, $\Pr_{x\leftarrow D_n}$ means that the probability is taken over x distributed according to D_n and $\Pr_A$ means that the probability is taken over the internal random bits of the algorithm A.

The algorithms that we study can output either 1 (accept) or 0 (reject), or $\perp$ (give up). They can also diverge, i.e., run forever (denoted ∞). For an algorithm A and an integer T, we denote by $A^{\leqslant T}$ the algorithm that behaves as A until the step T, and then gives up.

The *time* spent by a randomized algorithm A on input x is defined as the median time

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

We will also use a similar notation for the order statistics the "*probability p time*":

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

2.2. Randomized heuristic algorithms.

Definition 2.1. *$A(x,d)$ is a randomized heuristic algorithm for a distributional problem (L,D) if for every n,*

$$\Pr_{x\leftarrow D_n;A}[A(x,d) \neq L(x)] < \frac{1}{d}, \quad \textit{where} \quad L(x) = \begin{cases} 1, & x \in L, \\ 0, & x \notin L. \end{cases}$$

Remark 2.1. Note that [1] and [5] define randomized heuristic algorithms and acceptors in a different way separating the probabilities over x and over A. Note also that [5, Sect. 2] proves that algorithms defined in these two different ways simulate each other (the proof is given there for acceptors and goes for algorithms without changes).

Definition 2.2. *A function $f\colon \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ is* polynomially bounded on a set X *if there is a polynomial p such that for every $x \in X$ and $d \in \mathbb{N}$, $f(x,d) \leqslant p(|x|d)$.*

A heuristic algorithm A is polynomially bounded on set X *if its median time t_A is polynomially bounded on X. If X is equal to $\{0,1\}^*$ we omit it.*

Definition 2.3 ([1])**. HeurBPP** *is the class of distributional problems that can be solved by polynomially bounded randomized heuristic algorithms.*

Definition 2.4. *Function* $f\colon \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ dominates *function* $g\colon \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ *on set X (denoted $f \succeq g$), if there are polynomials p and q such that for all $x \in X$ and $d \in \mathbb{N}$,*

$$g(x,d) \leqslant \max_{d' \leqslant q(|x|d)} \{p(f(x,d')d|x|)\}.$$

Remark 2.2.

(1) If $f \succeq g$ on X and f is polynomially bounded on X, then so is g.
(2) $\succeq$ is transitive.

Definition 2.5. *For randomized heuristic algorithms A and A' for the same distributional problem (L,D), the algorithm A* simulates *A' if $t'_A \succeq t_A$ on* $\operatorname{supp} D = \{x \mid D(x) \neq 0\}$.

An optimal *randomized heuristic algorithm for a distributional problem (L,D) simulates every randomized heuristic algorithm for (L,D).*

2.3. Deterministic heuristic algorithms.

Definition 2.6. *A* deterministic *heuristic algorithm is a randomized heuristic algorithm that does not use its randomness.*

The running time t_A is now simply the number of steps made by the algorithm A. However, for deterministic heuristic algorithms, the notions of the polynomial boundness and the simulation will be relaxed by allowing the restrictions not to hold on a small number of inputs.

Definition 2.7. *A function $f\colon \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ is* polynomially bounded on the average *w.r.t. distribution D, if there is a polynomial p such that for every $n, d \in \mathbb{N}$,*

$$\Pr_{x \leftarrow D_n} [f(x,d) \leqslant p(n \cdot d)] \geqslant 1 - \frac{1}{2d}.$$

A deterministic heuristic algorithm is polynomially bounded on the average if its running time is polynomially bounded on the average.

Definition 2.8. *A function $f\colon \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$,* dominates *$g\colon \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$* on the average *w.r.t. distribution D (denoted $f \gtrsim g$), if there are*

polynomials p and q such that $q(n,d) \geqslant 2d$ and for every $n, d \in \mathbb{N}$,

$$\Pr_{x \leftarrow D_n}[g(x,d) \leqslant p(n \cdot d \cdot f(x, q(n,d)))] \geqslant 1 - \frac{1}{d}.$$

It is easy to see that the class of functions polynomially bounded on the average is closed under domination on the average.

Proposition 2.1. *Let $f \succsim g$ and f is polynomially bounded on the average w.r.t. D. Then g is also polynomially bounded on the average w.r.t. D.*

Proof. Let p and q be two polynomials in the definition of $\succsim$, and p' be a polynomial in the definition of polynomial boundness of g; without loss of generality we can assume that p is nondecreasing. The polynomial boundness and the restriction on q give

$$\Pr_{x \leftarrow D_n}[f(x, q(n,d)) \leqslant p'(n \cdot q(n,d))] \geqslant 1 - \frac{1}{q(n,d)} \geqslant 1 - \frac{1}{2d}.$$

Substituting it into the domination condition we get

$$\Pr_{x \leftarrow D_n}[g(x,d) \leqslant p(n \cdot d \cdot p'(n \cdot q(n,d)))] \geqslant 1 - \frac{1}{2d} - \frac{1}{2d} = 1 - \frac{1}{d}.$$

□

Definition 2.9. *For heuristic algorithms A and A' for a distributional problem (L, D), we say that A simulates A', if $t'_A \succsim t_A$ w.r.t. D.*

A deterministic heuristic algorithm for a distributional problem (L, D) is optimal on the average *if it simulates every other deterministic heuristic algorithm for (L, D).*

Definition 2.10 ([1])**. HeurP** *is the class of distributional problems that can be solved by deterministic heuristic algorithms that are polynomially bounded on the average.*

2.4. The problem of recognizing the image of an injective function. In this paper we concentrate on the following problem.

Definition 2.11. *Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable injective function such that The* problem of recognizing the image *is the distributional problem $(\operatorname{Im} f, U)$ where U is the uniform distribution. We will also denote by $\operatorname{Im} f$ the corresponding characteristic function, i.e., $(\operatorname{Im} f)(x) = 1$ if $x \in \operatorname{Im} f$ and $(\operatorname{Im} f)(x) = 0$ if $x \notin \operatorname{Im} f$.*

To show the importance of this problem and its nontriviality for heuristic algorithms let us consider a particularly hard case when f is a pseudorandom generator.

Definition 2.12 (see, e.g., [3, Sec. 3]). *Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function such that $|f(r)| = |r| + 1$ for all $r \in \{0,1\}^*$. Then f is called a pseudorandom generator if for every polynomial-time randomized algorithm A and for every polynomial p,*

$$\exists n_0\ \forall n > n_0\ \left| \Pr_{x \leftarrow U_n}[A(f(x)) = 1]\ -\ \Pr_{x \leftarrow U_{n+1}}[A(x) = 1] \right|\ <\ \frac{1}{p(n)}.$$

It is known that injective pseudorandom generators exist if one-way permutations exist [3].

Proposition 2.2 ([5, Theorem 5.2]). *If f is a pseudorandom generator, then there are no randomized heuristic algorithms for the problem* $(\operatorname{Im} f, U)$ *with running time that is polynomialy bounded on* $\overline{\operatorname{Im} f}$.

2.5. Estimator.

Definition 2.13. *We call an* estimator *an algorithm* Estimate$(A, x, g, v, \epsilon, T)$ *that given*

- *an algorithm A (i.e., its Goedel number), which can be either randomized or deterministic,*
- *an input $x \in \{0,1\}^n$,*
- *a function $S : \{0,1\}^n \to \{0,1\}^n$ (as an oracle),*
- *a value $v \in \{0,1\}$,*
- *a rational number $\epsilon \in (0;1)$,*
- *an integer T,*

runs in time upper bounded by a polynomial of T, n, and $\frac{1}{\epsilon}$ and outputs a rational number ρ such that

$$\Pr\left[\left|\rho\ -\ \Pr_{y \leftarrow g(U_n);A}[A^{\leqslant T}(y) = v]\right| \geqslant \epsilon\right]\ <\ \epsilon,$$

where the outermost probability is taken over the internal random bits of Estimate *and over uniformly distributed $x \in \{0,1\}^n$.*

Remark 2.3. At first glance, it may seem that the expected answer of Estimate is not related to x and therefore Estimate does not need x. However, we will see later that in the deterministic case the input x is the only source of pseudorandomness and thus it does matter for deterministic heuristic estimators. For the randomized case it can be indeed ignored.

Remark 2.4. In this paper, we use estimators for two functions: the identity function id and the function g that cuts the last bit of the input and applies the injective function f whose image we are trying to recognize, to the first $n-1$ bits of the input to get an n-bit string uniformly distributed on $\operatorname{Im} f \cap \{0,1\}^n$.

§3. The general construction of an optimal algorithm

In this section, we describe the "main" algorithm Opt for a distributional problem $(\operatorname{Im} f, U)$, which we use both in the deterministic and in the randomized case. It uses an enumerator $A_\bullet$ for algorithms of certain type (that is, A_i is a Turing machine with Goedel number i, and it can be either a randomized or a deterministic machine depending on the enumerator), and an estimator Estimate for the same type of algorithms. We assume that A_0 is a deterministic brute-force algorithm for testing the membership in $\operatorname{Im} f$ running in 2^{cn} steps for an integer constant $c \geqslant 1$ (note that $\operatorname{Im} f$ can be certainly accepted in time $O(2^n \cdot p(n))$, where $p(n)$ is the complexity of f).

The algorithm Opt resembles Levin's optimal algorithm for **NP** search problems [8]. It executes all algorithms A_i in parallel. When A_i stops with an answer v, the answer is verified in the same parallel process; if the verification is successful, Opt stops all parallel processes and outputs this answer. However, the verification procedure is very different from that of Levin (note that we have a single bit to verify). It depends on the answer: if $v = 0$, then we verify that A_i returns 0 on a randomly selected element of $\operatorname{Im} f$ with negligible probability. This is done by sampling elements from $\operatorname{Im} f$ and calculating the frequency of the answer 0 (note that the uniform distribution on the $\operatorname{Im} f$ is polynomial-time samplable using the function f itself). If $v = 1$, then we verify that A_i returns 1 on a randomly selected input uniformly distributed on the complement of $\operatorname{Im} f$ with negligible probability. However, the uniform distribution on $\overline{\operatorname{Im} f}$ may be hard to sample. Instead, we reduce this problem to the estimation of two other probabilities as

$$\Pr_{x \leftarrow U_n}[\dots] = \frac{1}{2} \Pr_{x \leftarrow U_n(\operatorname{Im} f)}[\dots] + \frac{1}{2} \Pr_{x \leftarrow U_n(\overline{\operatorname{Im} f})}[\dots].$$

Algorithm 3.1. $\operatorname{Opt}[A_\bullet, \operatorname{Estimate}](x, d)$

(1) Let $n = |x|$ and let $d' = 20cn^2 d$.

(2) For every $i \in \{0, 1, \ldots, n\}$, execute the following process in parallel:
- Run $A_i(x, d')$.
- If $i = 0$ and A_0 outputs $v \in \{0, 1\}$, then stop all parallel processes and output v.
- If $i > 0$ and $A_i(x, d')$ outputs $v \in \{0, 1\}$ in T steps, run $\mathrm{Test}(v, \mathrm{Estimate}, A_i, 2^{\lceil \log T \rceil}, x, d')$. If Test accepts, then stop all parallel processes and output v.

Algorithm 3.2. $\mathrm{Test}(v, \mathrm{Estimate}, A, T, x, d')$

(1) Let $\epsilon = \frac{2}{d'}$, $A'(x) = A(x, d')$ and
let $g(y \circ b) = f(y)$ for $y \in \{0, 1\}^{|x|-1}$, $b \in \{0, 1\}$.
(2) If $v = 0$:
(a) Compute $\rho = \mathrm{Estimate}(A', x, g, 0, \epsilon, T)$.
(b) If $\rho < 2\epsilon$, accept; otherwise reject.
(3) If $v = 1$:
(a) Compute $\alpha = \mathrm{Estimate}(A', x, g, 1, \epsilon, T)$.
(b) Compute $\beta = \mathrm{Estimate}(A', x, \mathrm{id}, 1, \epsilon, T)$.
(c) Accept, if $2\beta - \alpha < 4\epsilon$; otherwise reject.

In what follows $d' = 20cn^2d$ and $\epsilon = \frac{2}{d'} = \frac{1}{10cn^2d}$ as in the algorithms above.

Lemma 3.1. *For an algorithm A, denote* $\rho = \Pr_{x \leftarrow U_n(\mathrm{Im}\, f);A}[A^{\leqslant T}(x, d') = 1]$. *Let α and β be the random variables computed at step 3 of* $\mathrm{Test}(1, \mathrm{Estimate}, A, T, x, d')$. *Then* $\Pr[|\rho - (2\beta - \alpha)| \geqslant 3\epsilon] < 2\epsilon$.

Proof. Let

$$a = \Pr_{x \leftarrow f(U_{n-1});A}[A^{\leqslant T}(x, d) = 1] = \Pr_{x \leftarrow U_n(\mathrm{Im}\, f);A}[A^{\leqslant T}(x, d) = 1]$$

and let $b = \Pr_{x \leftarrow U_n;A}[A^{\leqslant T}(x, d) = 1]$. Clearly, $\rho = 2b - a$. By the definition of an estimator $\Pr[|a - \alpha| \geqslant \epsilon] < \epsilon$ and $\Pr[|b - \beta| \geqslant \epsilon] < \epsilon$. Using the triangle inequality we get $\Pr[|(2b - a) - (2\beta - \alpha)| \geqslant 3\epsilon] < 2\epsilon$. □

Theorem 3.1. *If* Estimate *is a randomized (resp., deterministic) estimator and $A_\bullet$ is an enumeration of randomized (resp., deterministic) Turing Mashines then* $\mathrm{Opt}[A_\bullet, \mathrm{Estimate}]$ *is a randomized (resp., deterministic) heuristic algorithm for* $(\mathrm{Im}\, f, U)$.

Proof. Since A_0 always gives the correct answer, it suffices to prove that for every $i \in \{1, 2, \ldots, n\}$

$$\Pr_{x \leftarrow U_n; A_i; \mathrm{Test}} \left[\begin{array}{l} A_i(x, d') \text{ outputs 0 or 1 in some } T \leqslant 2^{cn} \text{ steps } \wedge \\ A_i(x, d') \neq (\operatorname{Im} f)(x) \wedge \\ \mathrm{Test}(v, \mathrm{Estimate}, A_i, 2^{\lceil \log T \rceil}, x, d') = 1 \end{array} \right] < \frac{1}{dn}.$$

Since A_0 runs in 2^{cn} steps, no other algorithm A_i is allowed to run longer. Thus we can split every algorithm into cn "parts": $A_i^{\leqslant 1}$, $A_i^{\leqslant 2}$, $A_i^{\leqslant 4}$, ... and it now suffices to prove that for every $i \in \{1, 2, \ldots, n\}$, $k \in \{0, 1, \ldots, cn\}$

$$\Pr_{x \leftarrow U_n; A_i; \mathrm{Test}} \left[\begin{array}{l} A_i^{\leqslant 2^k}(x, d') \in \{0, 1\} \wedge \\ A_i^{\leqslant 2^k}(x, d') \neq (\operatorname{Im} f)(x) \wedge \\ \mathrm{Test}(v, \mathrm{Estimate}, A_i, 2^k, x, d') = 1 \end{array} \right] < \frac{1}{cdn^2}.$$

This probability can be split into two parts depending on the correct answer:

$$\frac{1}{2} \cdot \Pr_{x \leftarrow f(U_{n-1}); A_i} [A_i^{\leqslant 2^k}(x, d') = 0 \wedge \ldots] + \frac{1}{2} \cdot \Pr_{x \leftarrow U_n(\overline{\operatorname{Im} f}); A_i} [A_i^{\leqslant 2^k}(x, d') = 1 \wedge \ldots].$$

To bound the first part, note that if $\Pr_{x \leftarrow f(U_{n-1}); A_i} [A_i^{\leqslant 2^k}(x, d') = 0] > 3\epsilon$, then by the definitions of Estimate and Test we have $\Pr[\mathrm{Test}(0, \mathrm{Estimate}, A_i, 2^k, x, d') = 1] < \epsilon$. Thus the first part of the probability is less than $\frac{3}{2}\epsilon$.

We now consider the case when $x \leftarrow U_n(\overline{\operatorname{Im} f})$. By Lemma 3.1, if $\Pr[A_i^{\leqslant 2^k}(x, d') = 1] > 7\epsilon$, then $\Pr[\mathrm{Test}(1, \mathrm{Estimate}, A_i, 2^k, x, d') = 1] < 2\epsilon$. Thus the second part of the probability is less than $\frac{7}{2}\epsilon$. In total we have $\frac{3}{2}\epsilon + \frac{7}{2}\epsilon < \frac{1}{cdn^2}$ by the definition of ϵ and d'. □

Lemma 3.2. *Let A be a heuristic algorithm for $(\operatorname{Im} f, U)$. Then for every integer T and any $v \in \{0, 1\}$,*

$$\Pr_{x \leftarrow U_n; \mathrm{Test}} [\mathrm{Test}(v, \mathrm{Estimate}, A, T, x, d') = 0] < 2\epsilon.$$

Proof. Consider $v = 0$. Then Test rejects with probability

$$\Pr_{y \leftarrow f(U_{n-1}); A} [A^{\leqslant T}(y, d') = 0] < \frac{2}{d'} = \epsilon.$$

Consider now $v = 1$. Since, by Lemma 3.1,

$$\Pr_{x \leftarrow U_n(\overline{\operatorname{Im} f}); A} [A^{\leqslant T}(y, d') = 1] < \frac{2}{d'} = \epsilon,$$

Test rejects with probability less than 2ϵ. □

§4. An optimal randomized heuristic algorithm

In this section we describe the randomized estimator for randomized algorithms, which completes the construction of an optimal randomized heuristic algorithm. The algorithm we implement in the randomized estimator is very straightforward: execute $A^{\leqslant T}$ sufficiently many times on random inputs sampled using the function g evaluated at randomly generated points; then calculate the frequency of the correct answer.

Algorithm 4.1. Estimate-Random$(A, x, g, v, \epsilon, T)$

- Let $s = \left\lceil \frac{\ln(2/\epsilon)}{\epsilon^2} \right\rceil + 1$.
- For $i = 1, 2, \ldots, s$, do
  (1) Generate $y \leftarrow g(U_n)$.
  (2) Execute $A^{\leqslant T}(y)$; let $u_i = 1$ if the answer equals v, and let $u_i = 0$ otherwise.
- Output $\frac{1}{s} \sum_{i=1}^{s} u_i$.

Lemma 4.1. *The algorithm Estimate-Random is an estimator for randomized algorithms.*

Proof. By Chernoff bounds (see, e.g., [9]),

$$\Pr\left[\left|\frac{1}{s}\sum_{i=1}^{s} u_i - \Pr_{y \leftarrow g(U_n), A}[A^{\leqslant T}(y) = v]\right| \geqslant \epsilon\right] < 2\mathrm{e}^{-2\epsilon^2 s} < \epsilon.$$

□

Remark 4.1. In fact, a slightly stronger statement holds. Namely, the probability can be taken over internal random bits only and not also over the inputs as it is stated in the definition of an estimator.

Lemma 4.2. *For any randomized heuristic algorithm B for the problem $(\operatorname{Im} f, U)$, $t_B \succeq t^{(1/4)}_{\mathrm{Opt}[A_\bullet, \textit{Estimate-Random}]}$, where $A_\bullet$ enumerates all randomized algorithms.*

Proof. Let $B = A_i$. To show the asymptotic bound, it suffices to consider $|x| \geqslant i$. Then A_i is executed by Opt. Since Estimate-Random does not use x, Lemma 3.2 implies that for any x, $\Pr_{\mathrm{Test}}[\mathrm{Test}(v, \text{Estimate-Random}, A_i, T, x, d') = 0]$ is less than 2ϵ. Therefore, for every x, the algorithm Opt

stops in time polynomial in n, d, and the median time $t_{A_i}(x, d')$ with probability at least $\frac{1}{2} - 2\epsilon$. □

Theorem 4.1. *Let us $A_\bullet$ enumerate all randomized algorithms. Consider the algorithm $R_{\mathrm{Opt}}(x, d)$ that executes three parallel copies of* $\mathrm{Opt}[A_\bullet,$ *Estimate-Random*$](x, 3d)$ *run in parallel; the parallel execution is stopped as soon as one of the copies accepts or rejects. Then R_{Opt} is optimal randomized heuristic algorithm for* $(\mathrm{Im}\, f, U)$.

Proof. By theorem 3.1 algorithm R_{Opt} is a heuristic randomized algorithm for $(\mathrm{Im}\, f, U)$. With probability at least $\frac{1}{2}$ at least one of three parallel executions of $\mathrm{Opt}[A_\bullet, \text{Estimate-Random}](x, 3d)$ stops in at most $t^{(1/4)}_{\mathrm{Opt}[A_\bullet, \text{Estimate-Random}]}(x, 3d)$ steps, and $t^{(1/4)}_{\mathrm{Opt}[A_\bullet, \text{Estimate-Random}]} \succeq t_{R_{\mathrm{Opt}}}$. By Lemma 4.2, for any randomized heuristic algorithm B for the problem $(\mathrm{Im}\, f, U)$ the following is satisfied: $t_B \succeq t^{(1/4)}_{\mathrm{Opt}[A_\bullet, \text{Estimate-Random}]}$, therefore $t_B \succeq t_{R_{Opt}}$. □

§5. An optimal deterministic heuristic algorithm

To complete the construction of an optimal randomized deterministic algorithm we need a deterministic estimator. In contrast to the randomized case, we cannot generate truly random inputs for $A^{\leqslant T}$. Thus we replace them by pseudorandom inputs. In order to make this derandomization, we use the following result by Goldreich and Wigderson.

Theorem 5.1 ([4])**.** *There exists a positive constant γ such that for all integer n and for all $\delta \geqslant 2^{-\gamma n}$ there exists a family of functions $\mathcal{F}_{\delta,n}$, each mapping $\{0,1\}^n$ to itself, satisfying the following properties.*

- *Succinctness: there exists a bijection between $\{0,1\}^{l(\delta)}$ and $\mathcal{F}_{\delta,n}$, where $l(\delta) = O(\log \frac{1}{\delta})$. Let $\phi_{\alpha,n}$ denote the function from $\mathcal{F}_{\delta,n}$ corresponding to $\alpha \in \{0,1\}^{l(\delta)}$. This property means that the number of functions in the family $F_{\delta,n}$ is polynomial in $\frac{1}{\delta}$.*
- *Efficient evaluation: there exists a logspace algorithm that takes two inputs: $\alpha \in \{0,1\}^{l(\delta)}$, a string $x \in \{0,1\}^n$ and returns $\phi_{\alpha,n}(x)$.*
- *Mixing property: for every two subsets $A, B \subseteq \{0,1\}^n$ there exists $\mathcal{F}_{A,B,\delta,n} \subset \mathcal{F}_{\delta,n}$ such that $|\mathcal{F}_{A,B,\delta,n}| \geqslant (1-\delta)|\mathcal{F}_{\delta,n}|$ and for every function $\phi \in \mathcal{F}_{A,B,\delta,n}$:*

$$\left| \Pr_{x \leftarrow U_n} [x \in A \wedge \phi(x) \in B] - \rho(A)\rho(B) \right| \leqslant \delta,$$

where $\rho(S) = \frac{|S|}{2^n}$ *denotes the density of the set* S.

Corollary 5.1. *In terms of Theorem 5.1, for every two subsets* $A, B \subseteq \{0,1\}^n$,

$$\left| \Pr_{x \leftarrow U_n, \phi \leftarrow U(\mathcal{F}_{\delta,n})} [x \in A \wedge \phi(x) \in B] - \rho(A)\rho(B) \right| \leqslant 2\delta.$$

Proof.

$$\begin{aligned} &\Pr_{x \leftarrow U_n, \phi \leftarrow U(\mathcal{F}_{\delta,n})} [x \in A \wedge \phi(x) \in B] \\ &\quad = \Pr[x \in A \wedge \phi(x) \in B \mid \phi \in \mathcal{F}_{A,B,\delta,n}] \cdot \Pr[\phi \in \mathcal{F}_{A,B,\delta,n}] \\ &\quad + \Pr[x \in A \wedge \phi(x) \in B \mid \phi \notin \mathcal{F}_{A,B,\delta,n}] \cdot \Pr[\phi \notin \mathcal{F}_{A,B,\delta,n}]. \end{aligned}$$

Mixing property implies that the last quantity can be bounded from above by $\rho(A)\rho(B)+2\delta$, and from below by $(\rho(A)\rho(B)-\delta)(1-\delta) \geqslant \rho(A)\rho(B)-2\delta$, since $\rho(A)\rho(B) \leqslant 1$. □

We now describe a (deterministic) estimator for deterministic algorithms. Let $\mathcal{F}_{\delta,n}$ be the family of functions from Theorem 5.1.

Algorithm 5.1. Estimate-Deterministic$(A, x, g, v, \epsilon, T)$

- Let $n = |x|$ and $\delta = \frac{1}{8}\epsilon^2$.
- If $\delta < 2^{-\gamma n}$, then execute $A^{\leqslant T}(y)$ for every $y \in \{0,1\}^n$, compute the relative frequency of the answer v and output this number.
- If $\delta \geqslant 2^{-\gamma n}$, then for every $\phi \in \mathcal{F}_{\delta,n}$, execute $A^{\leqslant T}(g(\phi(x)))$, compute the relative frequency of the answer v and output this number.

Proposition 5.1. *The algorithm Estimate-Deterministic is an estimator.*

Proof. If $\delta < 2^{-\gamma n}$, the algorithm Estimate-Deterministic computes the exact answer, and it has enough time for that, because δ is so small.

Otherwise, let $B = \{y \in \{0,1\}^n \mid A^{\leqslant T}(g(y)) = v\}$. Let

$$C_+ = \{x \in \{0,1\}^n \mid \Pr_{\phi \leftarrow U(\mathcal{F}_{\delta,n})} [\phi(x) \in B] \geqslant \rho(B) + \epsilon\},$$

and let

$$C_- = \{x \in \{0,1\}^n \mid \Pr_{\phi \leftarrow U(\mathcal{F}_{\delta,n})} [\phi(x) \in B] \leqslant \rho(B) - \epsilon\}.$$

Let $\rho(C_+) \geqslant \frac{\epsilon}{2}$. Then

$$\Pr_{\substack{x \leftarrow U_n, \\ \phi \leftarrow U(\mathcal{F}_{\delta,n})}} [x \in C_+ \wedge \phi(x) \in B] \geqslant \rho(C_+)(\rho(B) + \epsilon) \geqslant \rho(C_+)\rho(B) + \frac{\epsilon^2}{2},$$

which contradicts Corollary 5.1. Therefore, $\rho(C_+)<\frac{\epsilon}{2}$. Similarly, $\rho(C_-)<\frac{\epsilon}{2}$. Thus $\rho(C_- \cup C_+) < \epsilon$. □

Theorem 5.2. *The algorithm* $\mathrm{Opt}[A_\bullet, \textit{Estimate-Deterministic}]$ *is optimal on the average, where* $A_\bullet$ *enumerates all deterministic algorithms.*

Proof. Let A_i be a (correct) deterministic heuristic algorithm for $(\operatorname{Im} f, U)$. To show the asymptotic bound, it suffices to consider $|x| \geqslant i$. Then the algorithm A_i is executed by Opt.

To estimate the fraction of the inputs x such that $\mathrm{Test}(v, \text{Estimate-Deterministic}, A_i, T, x, d')$ rejects, note that Estimate-Deterministic does not use randomness. Therefore Lemma 3.2 implies that this fraction is less than $2\epsilon < \frac{1}{2d}$.

For every other x, the running time of Opt is polynomial in n, d, and $t_{A_i}(x, d')$. □

Remark 5.1. The algorithm constructed in Theorem 5.2 is also an optimal-on-the-average deterministic *acceptor* for the distributional problem $(\operatorname{Im} f, U)$ as well as for the problem $(\overline{\operatorname{Im} f}, U)$. (We refer the reader to Section 6 for the precise definitions and statements.)

§6. Deterministic heuristic acceptors and proof systems

In this section, we give the definitions of deterministic heuristic acceptors and automatizable deterministic heuristic proof systems and prove that they are equivalent in terms of the running time vs proof length. While this is not difficult to see, it is in contrast with the situation in the randomized setting where only the equivalence to *weakly* automatizable proof systems is proved [5].

Note that [5] considers distributional proving problems, i.e., distributional problems (L, D) with $L \cap \operatorname{supp} D = \varnothing$. In this appendix we use a natural generalization of these definitions to arbitrary distributions in order to keep the same notation as in the main part of the paper.

Definition 6.1. *A deterministic heuristic acceptor for a distributed problem* (L, D) *is a deterministic algorithm* $A(x, d)$ *such that*

- *For every* x *and* d, *the algorithm* $A(x, d)$ *either does not stop or outputs* 1 *(i.e., accepts).*
- *For every* $x \in L$ *and* $d \in \mathbb{N}$, $A(x, d) = 1$.
- *For every* d, $\Pr_{x \leftarrow D_n}[x \notin L \wedge A(x, d) = 1] < \frac{1}{d}$.

Proposition 6.1. *Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable injective function such that $|f(x)| = |x|+1$. The problem $(\operatorname{Im} f, U)$ has a deterministic heuristic acceptor that is optimal on the average with respect to $f(U)$. Similarly the problem $(\overline{\operatorname{Im} f}, U)$ has a deterministic heuristic acceptor that is optimal on the average with respect to $U(\overline{\operatorname{Im} f})$.*

Proof. Note that by running a brute-force search in parallel we can transform an acceptor into an algorithm. If we run a brute-force search for a negative response we transform an algorithm into an acceptor (that does not err on the language). Then we can apply Theorem 5.2. □

Definition 6.2. *A deterministic heuristic proof system for a distributional problem (L, D) is a deterministic algorithm $\Pi(x, w, d)$ such that*

- *$\Pi(x, w, d)$ runs in time $(|x|d)^{O(1)}$.*
- *For every $x \in L$ and $d \in \mathbb{N}$, there exists $w \in \{0,1\}^*$ such that $\Pi(x, w, d) = 1$. We call such a string w a $\Pi^{(d)}$-*proof *of x.*
- *For every d, $\Pr_{x \leftarrow D_n}[x \notin L \wedge \exists w\ \Pi(x, w, d) = 1] < \frac{1}{d}$.*

If for $x \notin L$, there is a string w such that $\Pi(x, w, d) = 1$, we call w a fake $\Pi^{(d)}$-proof *of x.*

For $x \in L$, we denote by $\ell_\Pi(x, d)$ the length of the shortest $\Pi^{(d)}$-proof of x.

Definition 6.3. *A deterministic algorithm $B(x, d)$ is an* automatization procedure *for a heuristic deterministic proof system Π if for every $x \in L$ and $d \in \mathbb{N}$, the algorithm $B(x, d)$ takes time polynomial in $|x|$, d, and $\ell_\Pi(x, d)$ and outputs a $\Pi^{(d)}$-proof. For $x \notin L$, the behavior of the algorithm B is not restricted.*

A proof system is automatizable *if there is an automatization procedure for it.*

Similarly to the classical case, heuristic deterministic acceptors and proof systems can be converted into each other. The details follow.

Let $A(x, d)$ be a deterministic heuristic acceptor for a distributed problem (L, D). The corresponding proof system Π_A can be defined as follows: $\Pi_A(x, 1^T, d) = 1$ if $A(x, d)$ accepts in at most T steps. Clearly, Π_A is an automatizable heuristic proof system; the automatization procedure just simulates $A(x, d)$, computes the number of steps T required for the acceptance, and outputs 1^T. Also $\ell_{\Pi_A}(x, d) \leqslant (t_A(x, d) + |x| + d)^{O(1)}$.

Assume now that Π is an automatizable proof system for (L, D), and B is its automatization procedure. The corresponding acceptor $A_\Pi(x, d)$

can be defined as follows: simulate $B(x, d)$; if it outputs a proof, accept; otherwise do not stop.

Since these transformations translate the running time into the proof length and vice versa, we can avoid going into the details of specific heuristic simulations (similar to $\succeq$ we defined for heuristic algorithms) and prove a more general statement.

Definition 6.4. *Let $\prec$ be a transitive relation on the set of functions from $L \times \mathbb{N}$ to $\mathbb{N}$. We call it a* simulation *if for every two such functions f, g, if $f(x, d) \leqslant (g(x, d) + |x| + d)^{O(1)}$ then $f \prec g$.*

Proposition 6.2. *Let $\prec$ be a simulation. Then a distributed problem (L, D) has an acceptor A with the smallest t_A under $\prec$ (in the set of the running time functions for all possible acceptors) iff there is a deterministic heuristic automatizable proof system with the smallest ℓ_Π under $\prec$.*

Proof. We use the correspondence described above (A_Π and Π_A). Assume that A is an acceptor with the smallest running time t_A. Then Π_A is a proof system with the smallest ℓ_{Π_A}. Indeed, consider another proof system Π. The construction of A_Π implies that $t_{A_\Pi} \prec \ell_\Pi$. Since t_A is the smallest running time for acceptors, $t_A \prec t_{A_\Pi}$. The construction of Π_A implies that $\ell_{\Pi_A} \prec t_A$. By transitivity, $\ell_{\Pi_A} \prec \ell_\Pi$.

The proof of the converse is similar. □

Corollary 6.1. *The distributional problem $(\overline{\operatorname{Im} f}, U)$ has a deterministic heuristic automatizable proof system Π that is optimal on the average with respect to $f(U)$ (i.e., its length function ℓ_Π is the smallest under $\succeq$).*

§7. Further research

A natural question is to generalize the construction to suit any (not necessarily injective) function.

However, a much more challenging question is to construct an optimal *heuristic proof system* for $(\overline{\operatorname{Im} f}, U)$ (see [5] and Sec. 6 for the rigorous definition of a heuristic proof system).

References

1. A. Bogdanov, L. Trevisan, *Average-case complexity*, — Foundation and Trends in Theoretical Computer Science **2**, No. 1 (2006), 1–106.
2. Y. Chen, J. Flum, M. Müller, *Hard instances of algorithms and proof systems.* — Electronic Colloquium on Computational Complexity **11-085**, (2011).

3. O. Goldreich, *Foundation of Cryptography: Basic Tools.* Cambridge University Press (1995).
4. O. Goldreich, A. Wigderson, *Tiny families of functions with random properties: A quality-size trade-off for hashing.* — Random Structures Algorithms **11**, No. 4 (1997), 315–343.
5. E. A. Hirsch, D. Itsykson, I. Monakov, A Smal, *On optimal heuristic randomized semidecision procedures, with applications to proof complexity and cryptography.* — Theory of Computing Systems. To appear 2012.
6. E. A. Hirsch, *Optimal acceptors and optimal proof systems.* — TAMC, Lect. Notes Computer Sci. **6108** (2010), 28–39;
7. J. Krajíček, P. Pudlák, *Propositional proof systems, the consistency of first order theories and the complexity of computations.* — J. Symbolic Logic **54**, No. 3 (1989), 1063–1079.
8. L. A. Levin, *Universal sequential search problems.* — Problems Information Transmission **9** (1973), 265–266.
9. C. McDiarmid, *Concentration*, — Algorithms Combinatorics, Springer-Verlag **16** (1998), 195–248,
10. J. Messner, *On optimal algorithms and optimal proof systems.* In: Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science. Lect. Notes Computer Sci. **1563** (1999), 361–372.
11. H. Monroe, *Speedup for natural problems and noncomputability.* — Theor. Computer Sci. **412**, No. 4–5 (2011), 478–481.
12. Z. Sadowski, *On an optimal deterministic algorithm for SAT*, — In: Proceedings of CSL'98 **1584** Lect. Notes Computer Sci., Springer (1999), pp. 179-187.

Поступило 31 июля, 2011 г.

St.Petersburg Department
of the Steklov Mathematical Institute,
Fontanka 27,
St.Petersburg 191023,
Russia

E-mail: Web: http://logic.pdmi.ras.ru/{~hirsch,~dmitrits,~smal}

St.Petersburg Academic University,
Khlopina 8/3,
St.Petersburg 1945021, Russia