ON OPTIMAL PROBABILISTIC ALGORITHMS FOR SAT

YIJIA CHEN, JÖRG FLUM, AND MORITZ MÜLLER

1. Introduction. A major aim in the development of algorithms for hard problems is to decrease the running time. In particular one asks for algorithms that are optimal: A deterministic algorithm $\mathbb{A}$ deciding a language $L \subseteq \Sigma^*$ is *optimal* (or *(polynomially) optimal* or *p-optimal*) if for any other algorithm $\mathbb{B}$ deciding L there is a polynomial p such that

$$t_{\mathbb{A}}(x) \leq p(t_{\mathbb{B}}(x) + |x|) \tag{1}$$

for all $x \in \Sigma^*$. Here $t_{\mathbb{A}}(x)$ denotes the running time of $\mathbb{A}$ on input x. If (1) is only required for all $x \in L$, then $\mathbb{A}$ is said to be an *almost optimal algorithm for* L (or to be *optimal on positive instances of* L).

Various recent papers address the question whether such optimal algorithms exist for NP-complete or coNP-complete problems (cf. [1]), even though the problem has already been considered in the seventies when Levin [4] observed that there exists an optimal algorithm that finds a witness for every satisfiable propositional formula. Furthermore the relationship between the existence of almost optimal algorithms for a language L and the existence of "optimal" proof systems for L has been studied [3, 5].

Here we present a result (see Theorem 2.1) that can be interpreted as stating that (under the assumption of the existence of one-way functions) there is no optimal
probabilistic algorithm for SAT.

2. Probabilistic speed-up. For a propositional formula α we denote by $\|\alpha\|$ the number of literals in it, counting repetitions. Hence, the actual length of any reasonable encoding of α is polynomially related to $\|\alpha\|$.

Theorem 2.1. *Assume one-way functions exist. Then for every probabilistic algorithm $\mathbb{A}$ deciding* SAT *there exists a probabilistic algorithm $\mathbb{B}$ deciding* SAT *such that for all $d \in \mathbb{N}$ and sufficiently large $n \in \mathbb{N}$*

$$\Pr\Big[\text{there is a satisfiable } \alpha \text{ with } \|\alpha\| = n \text{ such that } \mathbb{A} \text{ does not accept } \alpha \text{ in at most } (t_{\mathbb{B}}(\alpha) + \|\alpha\|)^d \text{ steps}\Big] \geq \frac{1}{5}.$$

Note that $t_{\mathbb{A}}(\alpha)$ and $t_{\mathbb{B}}(\alpha)$ are random variables, and the probability is taken over the coin tosses of $\mathbb{A}$ and $\mathbb{B}$ on α.

Here we say that a probabilistic algorithm $\mathbb{A}$ decides SAT if it decides SAT as a nondeterministic algorithm, that is

$$\alpha \in \textsc{Sat} \Longrightarrow \Pr[\mathbb{A} \text{ accepts } \alpha] > 0,$$
$$\alpha \notin \textsc{Sat} \Longrightarrow \Pr[\mathbb{A} \text{ accepts } \alpha] = 0.$$

In particular, $\mathbb{A}$ can only err on 'yes'-instances.

Note that in the first condition the error probability is not demanded to be bounded away from 0, say by a constant $\varepsilon > 0$. As a more usual notion of probabilistic decision, say $\mathbb{A}$ *decides* SAT *with one-sided error* ε if

$$\alpha \in \textsc{Sat} \Longrightarrow \Pr[\mathbb{A} \text{ accepts } \alpha] > 1 - \varepsilon,$$
$$\alpha \notin \textsc{Sat} \Longrightarrow \Pr[\mathbb{A} \text{ accepts } \alpha] = 0.$$

For this concept we get

Corollary 2.2. *Assume one-way functions exist and let $\varepsilon > 0$. Then for every probabilistic algorithm $\mathbb{A}$ deciding* SAT *with one-sided error ε there exists a probabilistic algorithm $\mathbb{B}$ deciding* SAT *with one-sided error ε such that for all $d \in \mathbb{N}$ and sufficiently large $n \in \mathbb{N}$*

$$\Pr\Big[\textit{there is a satisfiable } \alpha \textit{ with } \|\alpha\| = n \textit{ such that } \mathbb{A} \textit{ does not accept } \alpha \textit{ in at most } (t_{\mathbb{B}}(\alpha) + \|\alpha\|)^d \textit{ steps}\Big] \geq \frac{1}{5}.$$

This follows from the fact that in the proof of Theorem 2.1 we choose the algorithm $\mathbb{B}$ in such way that on any input α the error probability of $\mathbb{B}$ on α is not worse than the error probability of $\mathbb{A}$ on α.

3. Witnessing failure. The proof of Theorem 2.1 is based on the following result.

Theorem 3.1. *Assume that one-way functions exist. Then there is a probabilistic polynomial time algorithm $\mathbb{C}$ satisfying the following conditions.*

(1) *On input $n \in \mathbb{N}$ in unary the algorithm $\mathbb{C}$ outputs with probability one a satisfiable formula β with $\|\beta\| = n$.*
(2) *For every $d \in \mathbb{N}$ and every probabilistic algorithm $\mathbb{A}$ deciding* SAT *and sufficiently large $n \in \mathbb{N}$*

$$\Pr\big[\mathbb{A} \textit{ does not accept } \mathbb{C}(n) \textit{ in } n^d \textit{ steps}\big] \geq \frac{1}{3}.$$

In the terminology of fixed-parameter tractability this theorem tells us that the parameterized construction problem associated with the following parameterized decision problem p-COUNTEREXAMPLE-SAT is in a suitably defined class of randomized nonuniform fixed-parameter tractable problems.

Instance:	An algorithm $\mathbb{A}$ *deciding* SAT and $d, n \in \mathbb{N}$ in unary.
Parameter:	$\|\mathbb{A}\| + d$.
Problem:	Does there exist a satisfiable CNF-formula α with $\|\alpha\| = n$ such that $\mathbb{A}$ does not accept α in n^d many steps?

Note that this problem is a promise problem. We can show:

Theorem 3.2. *Assume that one-way functions exist. Then the problem* p-COUNTEREXAMPLE-SAT *is nonuniformly fixed-parameter tractable.*[1]

This result is an immediate consequence of the following

Theorem 3.3. *Assume that one-way functions exist. For every infinite set* $I \subseteq \mathbb{N}$ *the problem*

SAT_I	
Instance:	A CNF-formula α with $\|\alpha\| \in I$.
Problem:	Is α satisfiable?

is not *in* PTIME.

We consider the construction problem associated with the decision problem p-COUNTEREXAMPLE-SAT, that is:

Instance:	An algorithm $\mathbb{A}$ *deciding* SAT and $d, n \in \mathbb{N}$ in unary.
Parameter:	$\|\mathbb{A}\| + d$.
Problem:	Construct a satisfiable CNF-formula α with $\|\alpha\| = n$ such that $\mathbb{A}$ does not accept α in n^d many steps, if one exists.

We do not know anything on its (deterministic) complexity; its nonuniform fixed-parameter tractability would rule out the existence of strongly almost optimal algorithms for SAT. By definition, an algorithm $\mathbb{A}$ deciding SAT is a *strongly almost optimal algorithm for* SAT if there is a polynomial p such that for any other algorithm $\mathbb{B}$ deciding SAT

$$t_{\mathbb{A}}(\alpha) \leq p(t_{\mathbb{B}}(\alpha) + |\alpha|)$$

for all $\alpha \in$ SAT. Then the precise statement of the result just mentioned reads as follows:

[1]This means, there is a $c \in \mathbb{N}$ such that for every algorithm $\mathbb{A}$ deciding SAT and every $d \in \mathbb{N}$ there is an algorithm that decides for every $n \in \mathbb{N}$ whether $(\mathbb{A}, d, n)$ is a positive instance of p-COUNTEREXAMPLE-SAT in time $O(n^c)$; here the constant hidden in $O(\ )$ may depend on $\mathbb{A}$ and d.

Proposition 3.4. *Assume that* $\mathrm{P} \neq \mathrm{NP}$. *If the construction problem associated with* p-COUNTEREXAMPLE-SAT *is nonuniformly fixed-parameter tractable, then there is no strongly almost optimal algorithms for* SAT.

4. Some Proofs. We now show how to use an algorithm $\mathbb{C}$ as in Theorem 3.1 to prove Theorem 2.1.

Proof of Theorem 2.1 from Theorem 3.1: Let $\mathbb{A}$ be an algorithm deciding SAT. We choose $a \in \mathbb{N}$ such that for every $n \geq 2$ the running time of the algorithm $\mathbb{C}$ (provided by Theorem 3.1) on input n is bounded by n^a. We define the algorithm $\mathbb{B}$ as follows:

$\mathbb{B}(\alpha)$ // $\alpha \in \mathrm{CNF}$

1. $\beta \leftarrow \mathbb{C}(\|\alpha\|)$.
2. **if** $\alpha = \beta$ **then** accept and halt.
3. **else** Simulate $\mathbb{A}$ on α.

Let $d \in \mathbb{N}$ be arbitrary. Set $e := d \cdot (a+2) + 1$ and fix a sufficiently large $n \in \mathbb{N}$. Let S_n denote the range of $\mathbb{C}(n)$. Furthermore, let $T_{n,\beta,e}$ denote the set of all strings $r \in \{0,1\}^{n^e}$ that do not determine a (complete) accepting run of $\mathbb{A}$ on β that consists in at most n^e many steps. Observe that a (random) run of $\mathbb{A}$ does not accept β in at most n^e steps if and only if $\mathbb{A}$ on β *uses* $T_{n,\beta,e}$, that is, its first at most n^e many coin tosses on input β are described by some $r \in T_{n,\beta,e}$. Hence by (2) of Theorem 3.1 we conclude

$$\sum_{\beta \in S_n} \big(\Pr[\beta = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \in T_{n,\beta,e}]\big) \geq \frac{1}{3}. \tag{2}$$

Let $\alpha \in S_n$ and apply $\mathbb{B}$ to α. If the execution of $\beta \leftarrow \mathbb{C}(\|\alpha\|)$ in Line 1 yields $\beta = \alpha$, then the overall running time of the algorithm $\mathbb{B}$ is bounded by $O\big(n^2 + t_{\mathbb{C}}(n)\big) = O(n^{a+1}) \leq n^{a+2}$ for n is sufficiently large. If in such a case a run of the algorithm $\mathbb{A}$ on input α uses an $r \in T_{n,\alpha,e}$, then it does not accept α in time $n^e = n^{(a+2)\cdot d+1}$ and hence not in time $(t_{\mathbb{B}}(\alpha) + \|\alpha\|)^d$. Therefore,

$$\begin{aligned}
&\Pr\Big[\text{there is a satisfiable } \alpha \text{ with } \|\alpha\| = n \text{ such that} \\
&\qquad \mathbb{A} \text{ does not accept } \alpha \text{ in at most } (t_{\mathbb{B}}(\alpha) + \|\alpha\|)^d \text{ steps}\Big] \\
&\geq 1 - \Pr\big[\text{for every input } \alpha \in S_n \text{ the algorithm } \mathbb{B} \text{ does not generate } \alpha \\
&\qquad\qquad \text{in Line 3, or } \mathbb{A} \text{ does not use } T_{n,\alpha,e}\big] \\
&= 1 - \prod_{\alpha \in S_n} \big((1 - \Pr[\alpha = \mathbb{C}(n)]) + \Pr[\alpha = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \notin T_{n,\alpha,e}]\big) \\
&= 1 - \prod_{\alpha \in S_n} \big(1 - \Pr[\alpha = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \in T_{n,\alpha,e}]\big)
\end{aligned}$$

$$\geq 1 - \left(\frac{\sum_{\alpha \in S_n} \left(1 - \Pr[\alpha = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \in T_{n,\alpha,e}]\right)}{|S_n|} \right)^{|S_n|}$$

$$= 1 - \left(1 - \frac{\sum_{\alpha \in S_n} \Pr[\alpha = \mathbb{C}(n)] \cdot \Pr_{r \in \{0,1\}^{n^e}}[r \in T_{n,\alpha,e}]}{|S_n|} \right)^{|S_n|}$$

$$\geq 1 - \left(1 - \frac{1}{3 \cdot |S_n|} \right)^{|S_n|}$$

$$\geq \frac{1}{5}. \qquad \square$$

Theorem 3.1 immediately follows from the following lemma.

Lemma 4.1. *Assume one-way functions exist. Then there is a randomized polynomial time algorithm $\mathbb{H}$ satisfying the following conditions.*

(H1) *Given $n \in \mathbb{N}$ in unary the algorithm $\mathbb{H}$ computes with probability one a* satisfiable CNF *α of size $\|\alpha\| = n$.*

(H2) *For every* probabilistic *algorithm $\mathbb{A}$ deciding* SAT *and every $d, p \in \mathbb{N}$ there exists an $n_{\mathbb{A},d,p} \in \mathbb{N}$ such that for all $n \geq n_{\mathbb{A},d,p}$*

$$\Pr\left[\mathbb{A} \textit{ accepts } \mathbb{H}(n) \textit{ in time } n^d\right] \leq \frac{1}{2} + \frac{1}{n^p},$$

where the probability is taken uniformly over all possible outcomes of the internal coin tosses of the algorithms $\mathbb{A}$ and $\mathbb{H}$.

(H3) *The cardinality of the range of (the random variable) $\mathbb{H}(n)$ is superpolynomial in n.*

Sketch of proof: We present the construction of the algorithm $\mathbb{H}$. By the assumption that one-way functions exist, we know that there is a pseudorandom generator (e.g. see [2]), that is, there is an algorithm $\mathbb{G}$ such that:

(G1) For every $s \in \{0,1\}^*$ the algorithm $\mathbb{G}$ computes a string $\mathbb{G}(s)$ with $|\mathbb{G}(s)| = |s| + 1$ in time polynomial in $|s|$.

(G2) For every probabilistic polynomial time algorithm $\mathbb{D}$, every $p \in \mathbb{N}$, and all sufficiently large $\ell \in \mathbb{N}$ we have

$$\left| \Pr_{s \in \{0,1\}^\ell}\left[\mathbb{D}(\mathbb{G}(s)) = 1\right] - \Pr_{r \in \{0,1\}^{\ell+1}}\left[\mathbb{D}(r) = 1\right] \right| \leq \frac{1}{\ell^p}$$

(In the above terms, the probability is also taken over the internal coin toss of $\mathbb{D}$.)

Let the language Q be the range of $\mathbb{G}$,

$$Q := \left\{\mathbb{G}(s) \;\middle|\; s \in \{0,1\}^*\right\}.$$

Q is in NP by (G1). Hence, there is a polynomial time reduction $\mathbb{S}$ from Q to SAT, which we can assume to be injective. We choose a constant $c \in \mathbb{N}$ such that $\|\mathbb{S}(r)\| \leq |r|^c$ for every $r \in \{0,1\}^*$. For every propositional formula β and every $n \in \mathbb{N}$ with $n \geq \|\beta\|$ let $\beta(n)$ be an equivalent propositional formula with $\|\beta(n)\| = n$. We may assume that $\beta(n)$ is computed in time polynomial in n.

One can check that the following algorithm $\mathbb{H}$ has the properties claimed in the lemma.

$\mathbb{H}(n)$ // $n \in \mathbb{N}$

1. $m \leftarrow \lfloor \sqrt[c]{n-1} \rfloor - 1$

2. Choose an $s \in \{0,1\}^m$ uniformly at random.

3. $\beta \leftarrow \mathbb{S}(\mathbb{G}(s))$.

4. Output $\beta(n)$

□

Acknowledgements. The authors thank the John Templeton Foundation for its support under Grant #13152, *The Myriad Aspects of Infinity.*

References

[1] O. Beyersdorff and Z. Sadowski. Characterizing the existence of optimal proof systems and complete sets for promise classes. Electronic Colloquium on Computational Complexity, Report TR09-081, 2009.

[2] O. Goldreich. *Foundations of Cryptography, Volume 1 (Basic Tools)*. Cambridge University Press, 2001.

[3] J. Krajicèk and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. Jour. Symb. Logic, 54(3):1063–1079, 1989.

[4] L. Levin. Universal search problems (in russian). Problemy Peredachi Informatsii 9(3):115-116, 1973.

[5] J. Messner. On optimal algorithms and optimal proof systems. STACS'99, LNCS 1563:541–550 1999.

Yijia Chen
Shanghai Jiaotong University
China.
E-mail address: yijia.chen@cs.sjtu.edu.cn

Jörg Flum
Albert-Ludwigs-Universität Freiburg
Germany.
E-mail address: joerg.flum@math.uni-freiburg.de

Moritz Müller
Centre de Recerca Matemàtica (CRM)
Spain.
E-mail address: mmueller@crm.cat