On an optimal deterministic algorithm for SAT

Zenon Sadowski

Institute of Mathematics,
University at Białystok
15-257 Białystok, ul. Akademicka 2, Poland
e-mail: sadowski@math.uwb.edu.pl

Abstract. J. Krajíček and P. Pudlák proved that an almost optimal deterministic algorithm for $TAUT$ exists if and only if there exists a p-optimal proof system for $TAUT$. In this paper we prove that an almost optimal deterministic algorithm for SAT exists if and only if there exists a p-optimal proof system for SAT. Combining Krajíček and Pudlák's result with our result we show that an optimal deterministic algorithm for SAT exists if and only if both p-optimal proof systems for $TAUT$ and for SAT exist.

1 Introduction

A deterministic algorithm recognizing SAT is optimal if no other algorithm recognizing SAT has more than a polynomial speed-up over its running time (see [5]). Two versions of optimality appear in Computational Complexity: Levin's optimality and Krajíček - Pudlák's optimality. In this paper we are mainly concerned with an optimal algorithm possessing Krajíček - Pudlák's optimality property. If the optimality property is stated only for any input string x which belongs to SAT and nothing is claimed for other x's, we name such an algorithm as an almost optimal deterministic algorithm for SAT.

A proof system for a language L is a polynomial time computable function whose range is L. This notion was defined by S. Cook and R. Reckhow in [3]. In order to compare the efficiency of different proof systems, they considered the notion of p-simulation. Intuitively a proof system h p-simulates a second one g if there is a polynomial time computable function t translating proofs in g into proofs in h. A proof system is called p-optimal for L when it p-simulates any proof system for L. The question whether a p-optimal proof system for $TAUT$ (the set of tautologies in Propositional Logic) exists, posed by J. Krajíček and P. Pudlák, is an important one in the field (see [5, 7, 4]).

The problem of the existence of an optimal deterministic algorithm for SAT has been considered as early as the beginning of the **NP**-era. In [6] L. Levin observed that there exists an almost optimal deterministic algorithm for the functional version of SAT (see [8]). Using Levin's construction we can also build an almost optimal deterministic algorithm for the decision version of SAT with the optimality property named by us as Levin's optimality (see [9]).

Nearly two decades later J. Krajíček and P. Pudlák considered the notion of an almost optimal deterministic algorithm for $TAUT$ (see [5]). Their conception of optimality differs from the one attributed to L. Levin and is named by us as Krajíček-Pudlák's optimality. They proved that an almost optimal deterministic algorithm for $TAUT$ exists if and only if there exists a p-optimal proof system for $TAUT$.

The question whether it is also possible to characterize an optimal deterministic algorithm for $TAUT$ (or equivalently an optimal deterministic algorithm for SAT) in p-optimal proof systems terms arises naturally from Krajíček and Pudlák's work. We answer this question. We prove that an almost optimal deterministic algorithm for SAT exists if and only if there exists a p-optimal proof system for SAT. Combining Krajíček and Pudlák's result with our result we show that an optimal deterministic algorithm for SAT exists if and only if both proof systems for $TAUT$ and for SAT exist.

In the last section we discuss the problem of the existence of an optimal deterministic algorithm for SAT with Levin's optimality property.

The inspiration to write this paper comes from [5, 4].

2 Preliminaries

We assume some familiarity with basic complexity theory, see [1]. The symbol Σ denotes, throughout the paper, a certain fixed finite alphabet. The set of all strings over Σ is denoted by $\Sigma^\star$. For a string x, $|x|$ denotes the length of x. The symbol FP denotes the class of functions that can be computed in polynomial time.

We use Turing machines (acceptors and transducers) as our basic computational model. For a deterministic Turing machine M and an input w TIME(M, w) denotes the computing time of M on w.

We consider clocked deterministic polynomial time Turing transducers (PTM for short). In the sequel we will not distinguish between a machine and its code. We impose some restrictions on our encoding of PTMs. From the code of any PTM we can detect easily (in polynomial time) the natural k such that $n^k + k$ is its polynomial time bound. We consider only languages over the alphabet Σ (this means that, e. g. , boolean formulas have to be suitably encoded). The symbol $TAUT$ denotes the set (of all encodings) of propositional tautologies, SAT denotes the set of all satisfiable propositional formulas and $UNSAT$ denotes the set of all unsatisfiable propositional formulas.

Finally, $< .,\ldots,. >$ denotes some standard polynomial time computable tupling function.

3 Optimal algorithms and p-optimal proof systems

The notion of an almost optimal deterministic algorithm for $TAUT$ was introduced by J. Krajíček and P. Pudlák. We propose to consider almost optimal and optimal deterministic algorithms for a language L with Krajíček-Pudlák's optimality property.

Definition 1. (cf. [5, 9]) An almost optimal deterministic algorithm for L is a deterministic Turing machine M which recognizes L and such that for every deterministic Turing machine M' which recognizes L there exists a polynomial p such that for every $x \in L$

$$TIME(M; x) \leq p(|x|, TIME(M'; x))$$

If we state the optimality conditions for both "yes" and "no" inputs we obtain the definition of an optimal deterministic algorithm for L.

Definition 2. An optimal deterministic algorithm for L is a deterministic Turing machine M which recognizes L and such that for every deterministic Turing machine M' which recognizes L there exists a polynomial p such that for any $x \in \Sigma^\star$

$$TIME(M; x) \leq p(|x|, TIME(M'; x))$$

Fact 1.
If an optimal deterministic algorithm for SAT exists then **P = NP** *if and only if this algorithm works in polynomial time.*

Fact 2. *The following statements are equivalent.*

(i) There exists an optimal deterministic algorithm for SAT.
(ii) There exists an optimal deterministic algorithm for $TAUT$.
(iii) There exists an optimal deterministic algorithm for $UNSAT$.

Fact 3. *The following statements are equivalent.*

(i) There exists an almost optimal deterministic algorithm for $TAUT$.
(ii) There exists an almost optimal deterministic algorithm for $UNSAT$.

A systematic study of complexity of proof systems for Propositional Logic was started by S. Cook and R. Reckhow in [3]. They introduced the abstract notion of a proof system for $TAUT$. The question of the existence of a polynomially bounded proof system for $TAUT$ is connected with the problem **NP=co-NP?** (see [3, 7]). The existence of an optimal proof system for $TAUT$ is an important open question introduced by J. Krajíček and P. Pudlák in [5].

J. Köbler and J. Messner defined the abstract notion of a proof system for a language L in the following way:

Definition 3. (see [4]) A proof system for L is a polynomial time computable function $h : \Sigma^\star \xrightarrow{onto} L$.

If $h(w) = x$ we say that w is a proof of x in h.

It follows from the above definition that for any proof system h there exists a PTM M_h which computes h. Proof systems for L can be treated as nondeterministic algorithms accepting L.

The notion of p-simulation is useful in comparing the strength of different proof systems for L.

Definition 4. (see [3, 4]) Let h, h' be two proof systems for L. We say that h p-simulates h' if there exists a polynomial time computable function $\gamma : \Sigma^\star \longrightarrow \Sigma^\star$ such that for every $x \in L$ and every $w \in \Sigma^\star$, if w is a proof of x in h', then $\gamma(w)$ is a proof of x in h.

Definition 5. (see [3, 4]) A proof system is p-optimal for L if it p-simulates any proof system for L.

4 Test formulas

In this section we construct boolean formulas which can be used to verify for a given PTM M and an input w that M on input w produces a satisfiable boolean formula. We use these formulas in the proof of Theorem 1.

Definition 6. We say that a PTM M behaves well on input w if M on w outputs a satisfiable boolean formula.

To any PTM M and any $w \in \Sigma^\star$ we shall assign the boolean formula $TEST_{M,w}$ such that:
The formula $TEST_{M,w}$ is satisfiable if and only if M behaves well on input w.

Our construction of the formula $TEST_{M,w}$ is adapted from Cook's proof that SAT is **NP**-complete (cf. [2]). Let S be a fixed nondeterministic Turing machine working in polynomial time and accepting SAT. Let M' be the Turing machine which on any input x runs M and then runs S on the output produced by M. The formula $TEST_{M,w}$ is just Cook's formula for the pair $< M', w >$.

From the construction of test formulas it follows that they possess the following properties:

(1) Global uniformity property
There exists a function $f \in FP$ such that for any PTM N with time bound $n^k + k$ and for any $w \in \Sigma^\star$

$$f(\langle N, w, O^{|w|^k+k}\rangle) = TEST_{N,w}$$

(2) Local uniformity property
Let M be any fixed PTM. There exists a function $f_M \in FP$ such that for any $w \in \Sigma^\star$

$$f_M(\langle M, w\rangle) = TEST_{M,w}$$

Let T be an almost optimal deterministic algorithm for SAT. The following lemma intuitively says that T easily accepts test formulas for any fixed proof system for SAT.

Let $k : \Sigma^\star \xrightarrow{onto} SAT$ be a proof system for SAT. Then there exists a certain PTM M with time bound $n^l + l$ and such that M computes k.

Lemma 1. *For every PTM M such that M is the machine of a certain proof system $k : \Sigma^\star \xrightarrow{onto} SAT$ there exists a polynomial p such that for every $w \in \Sigma^\star$ it holds: $TIME(T, TEST_{M,w}) \leq p(|w|)$.*

Proof. Since M is the machine of the proof system k, $TEST_{M,w}$ is a satisfiable boolean formula for every $w \in \Sigma^{\star}$. The machine M is fixed, so there exists a deterministic Turing machine which recognizes in polynomial time the language $\{TEST_{M,w} : w \in \Sigma^{\star}\}$. Combining this machine with the "brute force" algorithm for SAT we obtain the deterministic machine N recognizing SAT and such that the following condition holds: There exists a polynomial q such that for every $w \in \Sigma^{\star}$
$TIME(N, TEST_{M,w}) \leq q(|w|)$. The desired conclusion follows from the definition of an almost optimal deterministic algorithm for SAT.

5 Consistent Turing machines

The notion of a consistent Turing machine will be used in the proof of Theorem 1.

Let M be a deterministic Turing machine.

Definition 7. We say that M is consistent if M accepts only satisfiable boolean formulas (if M accepts w, then $w \in SAT$).

Fact 4. *For every consistent Turing machine M there exists the proof system $g_M : \Sigma^{\star} \xrightarrow{onto} SAT$ such that for every w accepted by M*
$g_M(x) = w$, where x is the computation of M accepting w.

The function g_M is the proof system for SAT generated by M. Because we impose conditions only on w's accepted by M, there are many proof systems generated by M.

6 Krajíček - Pudlák's optimality

In this section we prove the equivalence between the problem of the existence of an optimal deterministic algorithm for SAT and the problem of whether SAT and $TAUT$ have p-optimal proof systems. This is the main result of our paper.

Theorem 1. *Statements (i) - (ii) are equivalent.*

(i) There exists an almost optimal deterministic algorithm for SAT.
(ii) There exists a p-optimal proof system for SAT.

Proof. $(i) \rightarrow (ii)$

Let T be an almost optimal deterministic algorithm for SAT. We say that a string $v \in \Sigma^{\star}$ is in good form if

$$v = \langle M, w, O^{|w|^k+k}, Comp \rangle$$

where: M is a PTM with $n^k + k$ time bound, $w \in \Sigma^{\star}$, $O^{|w|^k+k}$ is the sequence of zeros, $Comp$ is the computation of T accepting the formula $TEST_{M,w}$.

Let us notice, that if v is in good form then $TEST_{M,w}$ is a satisfiable boolean formula, hence PTM M behaves well on input w producing a certain satisfiable boolean formula α. The string v is a natural candidate for the proof of the formula α in a p-optimal proof system for SAT.

Let α_0 be a certain fixed satisfiable boolean formula. We define $h : \Sigma^\star \longrightarrow \Sigma^\star$ in the following way:

$h(v) = \alpha$ if v is in good form ($v = \langle M, w, O^{|w|^k+k}, Comp\rangle$) and α is a boolean formula produced by M on input w, otherwise $h(v) = \alpha_0$.

Clearly, $h : \Sigma^\star \overset{onto}{\longrightarrow} SAT$. It follows from global uniformity condition from Section 4 that $h \in FP$.This proves that h is a proof system for SAT.

We have to show that h p-simulates any proof system for SAT. Let g be a proof system for SAT computed by PTM M with time bound $n^k + k$.

The function $t : \Sigma^\star \longrightarrow \Sigma^\star$

$$t(x) = \langle M, x, O^{|x|^k+k}, Comp\rangle$$

translates proofs in g into proofs in h.

The word $Comp$ in the definition of t is the computation of T acccepting the formula $TEST_{M,x}$. It follows from local uniformity property and Lemma 1 (see Section 4) that $Comp$ can be constructed from x in polynomial time. This proves that $t \in FP$.

$(ii) \rightarrow (i)$ Our proof technique is adapted from [5].

Let Opt be the machine of a p-optimal proof system for SAT (Opt works in polynomial time). Let M_0, M_1, M_2, M_3, ... be an enumeration of all deterministic Turing machine acceptors such that M_0 is the trivial "brute force" algorithm recognizing SAT. Let T_1, T_2, T_3, ... be an enumeration of all Turing machine transducers.

The desired machine T, which is an almost optimal deterministic algorithm for SAT, is constructed as follows. It has two distinguished worktapes QUEUE FOR VERIFICATION and PARTIAL RESULTS. These tapes are blanked at first.

On an input w, $|w| = n$, T simulates the work of M_0, M_1, M_2, M_3, ..., M_n and T_1, T_2, T_3, ... T_n in several rounds. At the m-th round the machine T performs the following three groups of operations:

1. One additional computational step of M_0, M_1, M_2, M_3, ..., M_n on w. The histories of computations of M_1, M_2, M_3, ..., M_n are stored on the PARTIAL RESULTS tape, so performing computational steps of M_1, M_2, M_3, ..., M_n, the machine T updates the contents of the PARTIAL RESULTS tape. If a certain machine M_i, $1 \le i \le n$, accepts w and x is the computation of M_i accepting w, then $\langle M_i, x\rangle$ is stored on the QUEUE FOR VERIFICATION tape ($\langle M_i, x\rangle$ is added to QUEUE FOR VERIFICATION).
2. One additional computational step of $T_1, T_2, T_3, \ldots, T_n$ on every input x such that $\langle M_i, x\rangle$ for a certain M_i is stored on QUEUE FOR VERIFICATION at that time.
3. T checks whether there are strings x, y and integers $i, k \le n$ such that:

(a) M_i has accepted w and $\langle M_i, x\rangle$ is on the QUEUE FOR VERIFICATION tape.
(b) T_k has produced on input x a string y.
(c) y is a proof of w in Opt.
If this is the case, T halts and accepts w, otherwise it continues operating till M_0 halts and delivers YES or NO result.

First we shall show that T recognizes SAT. If T finishes the simulation of M_0, then this is clear. So suppose that M accepts w because the situation in (3) occurs. Since w has an Opt-proof, w is a satisfiable boolean formula.

Let N be any deterministic Turing machine which recognizes SAT. $N = M_i$ for a certain i. Since N is consistent, there exists a proof system generated by N, and there exists j such that T_j translates proofs in this system into Opt-proofs. There exists also a polynomial p bounding the running time of T_j.

Let w be any input such that $|w| \geq \max\{i, j\}$ and $w \in SAT$. Let $k = TIME(M_i, w)$. T accepts w in $p(k) + k$ round or sooner. Since the mth round of T takes polynomially many steps in n and m, there is a polynomial q such that $TIME(T, w) \leq q(n, k)$. This proves that T is almost optimal deterministic algorithm for SAT.

The following result was proved in [5].

Theorem 2. *(Krajíček, Pudlák) The following statements are equivalent.*
(i) There exists a p-optimal proof system for $TAUT$.
(ii) There exists an almost optimal deterministic algorithm for $TAUT$.

Using Theorem 1, Theorem 2, Fact 2, and Fact 3 we can prove our main theorem.

Theorem 3. *The following statements are equivalent.*
(i) $TAUT$ and SAT have p-optimal proof systems.
(ii) There exists an optimal deterministic algorithm for SAT.
(iii) There exists an optimal deterministic algorithm for $TAUT$.

7 Levin's optimality

The following theorem is attributed to L. A. Levin in [9].

Theorem 4. *(Levin)*
There exists a deterministic Turing machine L which recognizes SAT and such that for every deterministic Turing machine N which recognizes SAT there exists a polynomial p such that for any $w \in SAT$

$$TIME(L, w) \leq p(|w|, \max\{TIME(N, v) : v \in SAT, |v| \leq |w|\})$$

The machine L is named by us as an almost optimal deterministic algorithm for SAT with Levin's optimality property. We propose to introduce the following notion.

Definition 8. An optimal deterministic algorithm for SAT with Levin's optimality property is a deterministic Turing machine M which recognizes SAT and such that for every deterministic Turing machine M' which recognizes SAT there exists a polynomial p such that for any $w \in \Sigma^\star$

$$TIME(M,w) \leq p(|w|, max\{TIME(M',v) : \ |v| \leq |w|\})$$

It follows from Theorem 2, Fact 3 and Theorem 4 that the existence of a p-optimal proof system for $TAUT$ implies the existence of an optimal deterministic algorithm for SAT with Levin's optimality property.

References

1. Balcazar, J.L., Díaz, J., Gabarró, J.: Structural complexity I. 2nd edn. Springer-Verlag, Berlin Heidelberg New York (1995)
2. Cook, S.A.: The complexity of theorem proving procedures. In: Proc. 3rd ACM Symposium on Theory of Computing. (1971) 151–158
3. Cook, S.A., Reckhow R.A.: The relative efficiency of propositional proof systems. J. Symbolic Logic 44 (1979) 36–50
4. Köbler, J., Messner, J.: Complete Problems for Promise Classes by Optimal Proof Systems for Test Sets. To appear in Proceedings of the Computational Complexity Conference. (1998)
5. Krajíček, J., Pudlák, P.: Propositional proof systems, the consistency of first order theories and the complexity of computations. J. Symbolic Logic 54 (1989) 1063–1079
6. Levin, L.A.: Universal sorting problems. Problems of Information Transmission 9 (1973) 265 –266
7. Messner, J., Torán, J.: Optimal proof systems for Propositional Logic and complete sets. Electronic Colloquium on Computational Complexity. (1997) available via http://www.eccc.uni-trier.de/eccc/
8. Trakhtenbrot, B.A.: A survey of Russian Approaches to Perebor (Brute-Force Search) Algorithms. Annals of the History of Computing 6 (1984) 384–400
9. Verbitskii, O.V.: Optimal algorithms for co-NP sets and the EXP=NEXP problem. Matematicheskie Zametki M.V. Lomonosov Moscow State University 50 (1991) 37–46