

Hard instances of algorithms and proof systems

Yijia Chen¹, Jörg Flum², and Moritz Müller³

¹ Shanghai Jiaotong University, China, yijia.chen@cs.sjtu.edu.cn

² Universität Freiburg, Germany, joerg.flum@math.uni-freiburg.de

³ Centre de Recerca Matemàtica, Barcelona, Spain, mmueller@crm.cat

Abstract. Assuming that the class TAUT of tautologies of propositional logic has no almost optimal algorithm, we show that every algorithm \mathbb{A} deciding TAUT has a polynomial time computable sequence witnessing that \mathbb{A} is not almost optimal. The result extends to every Π_t^P -complete problem with $t \geq 1$; however, we show that assuming the Measure Hypothesis there is a problem which has no almost optimal algorithm but has an algorithm without hard sequences.

1. Introduction

Let \mathbb{A} be an algorithm deciding a problem Q . A sequence $(x_s)_{s \in \mathbb{N}}$ of strings in Q is *hard for \mathbb{A}* if it is computable in polynomial time and the sequence $(t_{\mathbb{A}}(x_s)_{s \in \mathbb{N}})$ is not polynomially bounded in s .⁴ Here, $t_{\mathbb{A}}(x)$ denotes the number of steps the algorithm \mathbb{A} takes on input x . Clearly, if \mathbb{A} is polynomial time, then \mathbb{A} has no hard sequences. Furthermore, an almost optimal algorithm for Q has no hard sequences either. Recall that an algorithm \mathbb{A} is *almost optimal for Q* if for every input $x \in Q$ the running time $t_{\mathbb{A}}(x)$ is polynomially bounded in $t_{\mathbb{B}}(x)$ for any other algorithm \mathbb{B} deciding Q . In fact, if $(x_s)_{s \in \mathbb{N}}$ is a hard sequence for an algorithm, then one can polynomially speed up it on $\{x_s \mid s \in \mathbb{N}\}$, so it cannot be almost optimal.

Central to this paper is the question: To what extent can we show that algorithms which are not almost optimal have hard sequences? Our main result states:

- (a) *If a coNP-complete problem Q has no almost optimal algorithm, then every algorithm deciding Q has hard sequences.*

Perhaps one would expect that one can strengthen (a) and show that even if a coNP-complete problem Q has an almost optimal algorithm, then every algorithm, which is not almost optimal and decides Q , has a hard sequence. However, we show:

If the Measure Hypothesis holds, then every coNP-complete problem with padding and with an almost optimal algorithm has an algorithm which is not almost optimal but has no hard sequences.

Even though we can extend the result (a) to Π_t^P -complete problems (with $t \geq 1$), apparently there are some limitations as we derive the following result:

If the Measure Hypothesis holds, then there is a problem Q which has no almost optimal algorithm but has an algorithm without hard sequences.

⁴ All notions will be defined in a precise manner later.

In particular, there are algorithms deciding such a Q and polynomially speeding up a given algorithm. That is, this notion of speeding up (e.g. considered in [13,9]) differs from our notion of the existence of a hard sequence.

Assume that a coNP-complete problem Q has no almost optimal algorithm. Can we even effectively assign to every algorithm deciding Q a hard sequence? We believe that under reasonable complexity-theoretic assumptions one should be able to show that such an effective procedure or at least a polynomial time procedure does not exist, but we were not able to show it. However, recall that by a result due to McCreight and Meyer [9] and redicovered by Messner [11] we know:

For every EXP-hard problem Q there is a polynomial time effective procedure assigning to every algorithm solving Q a sequence hard for it.

Hence, if $\text{EXP} = \text{NP}$, then for every NP-hard (and hence for every coNP-hard) problem Q there is a polynomial time effective procedure assigning a hard sequence to every algorithm deciding Q .

Our proof of (a) generalizes to nondeterministic algorithms. This “nondeterministic statement” yields a version of a result due to Krajíček which he derived for non-optimal propositional proof systems: If TAUT, the set of tautologies of propositional logic, has no optimal proof system, then for every propositional proof system \mathbb{P} there is a polynomial time computable sequence $(\alpha_s)_{s \in \mathbb{N}}$ of propositional tautologies α_s with $s \leq |\alpha_s|$ which only have superpolynomial \mathbb{P} -proofs. While it is well-known that nondeterministic algorithms for TAUT and propositional proof systems are more or less the same (so that the nondeterministic version of (a) essentially is Krajíček’s result), the relationship between deterministic algorithms deciding TAUT and propositional proof systems is more subtle. Nevertheless, we are able to use (a) to derive a statement on hard sequences for propositional proof systems in case that TAUT has no *polynomially* optimal proof system.

As a byproduct, we obtain results in “classical terms” for which we do not know proofs avoiding the machinery we develop here; for example, we get:

Let Q be coNP-complete. Then, Q has an almost optimal algorithm if and only if Q has a polynomially optimal proof system.

If TAUT has no almost optimal algorithm, then every coNP-hard problem has no almost optimal algorithm.

It is still open whether there exist problems outside of NP with optimal proof systems. We show their existence (in NE) assuming the Measure Hypothesis. Krajíček and Pudlák [7] proved that $\text{E} = \text{NE}$ implies that TAUT has an optimal proof system.

If for an algorithm \mathbb{A} deciding a problem Q we have a hard sequence $(x_s)_{s \in \mathbb{N}}$ satisfying $s \leq |x_s|$, then $\{x_s \mid s \in \mathbb{N}\}$ is a *hard set for \mathbb{A}* , that is, a polynomial time decidable subset of Q on which \mathbb{A} is not polynomial time. Messner [11] has shown for any Q with padding that all algorithms deciding Q have hard sets if and only if Q has no polynomially optimal proof system. We show for arbitrary Q that the existence of hard sets for all algorithms is equivalent to the existence of an effective enumeration

of all polynomial time decidable subsets of Q , a property which has turned out to be useful in various contexts (cf. [12, 2, 3]). We analyze what Messner’s result means for proof systems.

The content of the sections is the following. In Section 2 we recall some concepts. We deal with hard sequences for algorithms in Section 3 and for proof systems in Section 4. Section 5 is devoted to hard sets and Section 6 contains the results and the examples of problems with special properties obtained assuming that the Measure Hypothesis holds. Finally Section 7 gives an effective procedure yielding hard sequences for nondeterministic algorithms for coNEXP-hard problems.

2. Preliminaries

We denote by Σ the alphabet $\{0, 1\}$ and by $|x|$ the length of a string $x \in \Sigma^*$. We identify problems with subsets of Σ^* . *In this paper we always assume that Q denotes a decidable and nonempty problem.*

We denote by P (NP) the class of problems Q such that $x \in Q$ is solvable by a deterministic (nondeterministic) Turing machine in $|x|^{O(1)}$ steps (formally, $n^{O(1)}$ denotes the class of polynomially bounded functions on the natural numbers). A problem $Q \subseteq \Sigma^*$ has padding if there is a function $pad : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ computable in logarithmic space having the following properties:

- For any $x, y \in \Sigma^*$, $|pad(x, y)| > |x| + |y|$ and $(pad(x, y) \in Q \iff x \in Q)$.
- There is a logspace algorithm which, given $pad(x, y)$ recovers y .

By $\langle \dots, \dots \rangle$ we denote some standard logspace computable tupling function with logspace computable inverses.

If \mathbb{A} is a deterministic or nondeterministic algorithm and \mathbb{A} accepts the string x , then we denote by $t_{\mathbb{A}}(x)$ the minimum number of steps of an accepting run of \mathbb{A} on x ; if \mathbb{A} does not accept x , then $t_{\mathbb{A}}(x)$ is not defined. By $L(\mathbb{A})$ we denote the language accepted by \mathbb{A} . We use deterministic and nondeterministic Turing machines with Σ as alphabet as our basic computational model for algorithms (and we often use the notions “algorithm” and “Turing machine” synonymously). If necessary we will not distinguish between a Turing machine and its code, a string in Σ^* . *By default, algorithms are deterministic.* If an algorithm \mathbb{A} on input x eventually halts and outputs a value, we denote it by $\mathbb{A}(x)$.

3. Hard sequences for algorithms

In this section we derive the results concerning the existence of hard sequences for coNP-complete problems.

Let $Q \subseteq \Sigma^*$. A deterministic (nondeterministic) algorithm \mathbb{A} deciding (accepting) Q is *almost optimal* if for every deterministic (nondeterministic) algorithm \mathbb{B} deciding (accepting) Q we have

$$t_{\mathbb{A}}(x) \leq (t_{\mathbb{B}}(x) + |x|)^{O(1)}$$

for all $x \in Q$. Note that nothing is required for $x \notin Q$.

Clearly, every problem in P has an almost optimal algorithm and every problem in NP has an almost optimal nondeterministic algorithm. There are problems outside P

with an almost optimal algorithm (see Messner[11, Corollary 3.33]; we slightly improve his result in Section 6). However, it is not known whether there are problems outside NP having an almost optimal nondeterministic algorithm and it is not known whether there are problems with padding outside P having an almost optimal algorithm. We show in Section 6 that the former is true if the Measure Hypothesis holds.

We introduce the notion of hard sequence.

Definition 1. Let $Q \subseteq \Sigma^*$.

- (1) Let \mathbb{A} be a deterministic (nondeterministic) algorithm deciding (accepting) Q . A sequence $(x_s)_{s \in \mathbb{N}}$ is *hard for \mathbb{A}* if $\{x_s \mid s \in \mathbb{N}\} \subseteq Q$, the function $1^s \mapsto x_s$ is computable in polynomial time, and $t_{\mathbb{A}}(x_s)$ is not polynomially bounded in s .
- (2) The problem Q has *hard sequences for algorithms* if every algorithm deciding Q has a hard sequence.
- (3) The problem Q has *hard sequences for nondeterministic algorithms* if every nondeterministic algorithm accepting Q has a hard sequence.

The proof of the following lemma is straightforward; it shows that if $(x_s)_{s \in \mathbb{N}}$ is hard for an algorithm \mathbb{A} , then \mathbb{A} can be polynomially speeded up on $\{x_s \mid s \in \mathbb{N}\}$; thus \mathbb{A} can't be almost optimal.

Lemma 2. *Let \mathbb{A} be a deterministic (nondeterministic) algorithm deciding (accepting) Q . If \mathbb{A} has a hard sequence, then \mathbb{A} is not almost optimal.*

Proof. We prove the deterministic case, the nondeterministic case is obtained by the obvious modifications. So assume that the algorithm \mathbb{A} decides Q and has a hard sequence $(x_s)_{s \in \mathbb{N}}$; in particular,

$$t_{\mathbb{A}}(x_s) \text{ is not polynomially bounded in } s. \quad (1)$$

Let \mathbb{G} be a polynomial time algorithm computing the function $1^s \mapsto x_s$. The following algorithm \mathbb{G}^* accepts the set $\{x_s \mid s \in \mathbb{N}\}$ and for $x = x_s$ runs in time polynomial in s .

```

 $\mathbb{G}^*$  //  $x \in \Sigma^*$ 
1.  $\ell \leftarrow 0$ 
2. for  $s = 0$  to  $\ell$ 
3.     simulate the  $(\ell - s)$ th step of  $\mathbb{G}$  on  $1^s$ 
4.     if this simulation outputs  $y$  and  $y = x$  then accept and halt
5.  $\ell \leftarrow \ell + 1$ 
6. goto 2.

```

We consider the algorithm $\mathbb{A} \parallel \mathbb{G}^*$ that on input x runs \mathbb{A} and \mathbb{G}^* in parallel, both on input x , and halts, when the first of these algorithms halts, then answering in the same way. Hence, $\mathbb{A} \parallel \mathbb{G}^*$ accepts Q and $t_{\mathbb{A} \parallel \mathbb{G}^*}(x_s)$ is polynomially bounded in s . As $|x_s| \leq s^{O(1)}$, by (1) we see that $t_{\mathbb{A}}(x_s)$ is not polynomially bounded in $t_{\mathbb{A} \parallel \mathbb{G}^*}(x_s) + |x_s|$; thus $\mathbb{A} \parallel \mathbb{G}^*$ witnesses that \mathbb{A} is not an almost optimal algorithm. \square

We state the main result of this section (Remark 7 contains extensions of the result to further classes of problems Q). As already remarked in the Introduction part (b) of this theorem is a straightforward consequence of the corresponding result for propositional proof systems due to Krajíček.

Theorem 3. *Let Q be a coNP-complete problem. Then:*

- (a) Q has no almost optimal algorithm $\iff Q$ has hard sequences for algorithms.
- (b) Q has no almost optimal nondeterministic algorithm $\iff Q$ has hard sequences for nondeterministic algorithms.

The proofs of the implications from right to left are clear by the previous lemma. The following considerations will yield a proof of the converse direction. For a nondeterministic algorithm \mathbb{A} and $s \in \mathbb{N}$ let \mathbb{A}^s be the algorithm that rejects all $x \in \Sigma^*$ with $|x| > s$. If $|x| \leq s$, then it simulates s steps of \mathbb{A} on input x ; if this simulation halts and accepts, then \mathbb{A}^s accepts; otherwise it rejects.

Recall that by $L(\mathbb{A})$ we denote the language accepted by \mathbb{A} . For $Q \subseteq \Sigma^*$ we consider the *deterministic (nondeterministic) algorithm subset problem* $\text{DAS}(Q)$ ($\text{NAS}(Q)$)

$\text{DAS}(Q)$

Instance: A deterministic algorithm \mathbb{A} and 1^s with $s \in \mathbb{N}$.

Question: $L(\mathbb{A}^s) \subseteq Q$?

$\text{NAS}(Q)$

Instance: A nondeterministic algorithm \mathbb{A} and 1^s with $s \in \mathbb{N}$.

Question: $L(\mathbb{A}^s) \subseteq Q$?

The following two lemmas relate the equivalent statements in Theorem 3 (a) (in Theorem 3 (b)) to a statement concerning the complexity of $\text{DAS}(Q)$ (of $\text{NAS}(Q)$).

- Lemma 4.** (a) *If $\langle \mathbb{A}, 1^s \rangle \in \text{DAS}(Q)$ is solvable in time $s^{f(\mathbb{A})}$ for some function f , then Q has an almost optimal algorithm.*
- (b) *If there is a nondeterministic algorithm \mathbb{V} accepting $\text{NAS}(Q)$ such that for all $\langle \mathbb{A}, 1^s \rangle \in \text{NAS}(Q)$ we have $t_{\mathbb{V}}(\langle \mathbb{A}, 1^s \rangle) \leq s^{f(\mathbb{A})}$ for some function f , then Q has an almost optimal nondeterministic algorithm.*

Proof. Again we only prove (a). Let \mathbb{V} be an algorithm deciding $\langle \mathbb{A}, 1^s \rangle \in \text{DAS}(Q)$ in time $s^{f(\mathbb{A})}$ for some function f . Further let \mathbb{Q} be an algorithm deciding Q and let $\mathbb{A}_0, \mathbb{A}_1, \dots$ be an effective enumeration of all algorithms. Consider the following algorithm \mathbb{A} deciding Q .

<pre> A // $x \in \Sigma^*$ 1. simulate \mathbb{Q} on x and in parallel do the following 2. for $i = 0$ to x do in parallel 3. simulate \mathbb{A}_i on x 4. if \mathbb{A}_i accepts then 5. $s \leftarrow \max\{ x , \text{length of the run accepting } x\}$ 6. if \mathbb{V} accepts $\langle \mathbb{A}_i, 1^s \rangle$ then accept and halt 7. else never halt 8. else never halt 9. if \mathbb{Q} stops first then answer accordingly and halt. </pre>

It is easy to see that \mathbb{A} decides Q . We show it is almost optimal. Let \mathbb{B} be any algorithm deciding Q . We choose $i_{\mathbb{B}} \in \mathbb{N}$ such that $\mathbb{B} = \mathbb{A}_{i_{\mathbb{B}}}$. Note that \mathbb{V} accepts $\langle \mathbb{B}, 1^s \rangle$ for all s . Hence for inputs $x \in Q$ with $|x| \geq i_{\mathbb{B}}$ the algorithm \mathbb{A} , for $i = i_{\mathbb{B}}$, accepts x in Line 6 if it was not already accepted earlier. Thus, $t_{\mathbb{A}}(x)$ is polynomially bounded in

$$|x| + t_{\mathbb{B}}(x) + t_{\mathbb{V}} \left(\langle \mathbb{B}, 1^{\max\{|x|, t_{\mathbb{B}}(x)\}} \rangle \right),$$

where the term $t_{\mathbb{B}}(x)$ takes care of line 3. Hence, by assumption, it is polynomially bounded in $|x| + \max\{|x|, t_{\mathbb{B}}(x)\}^{f(\mathbb{B})}$. Altogether, $t_{\mathbb{A}}(x) \leq (|x| + t_{\mathbb{B}}(x))^{O(1)}$. \square

If Q is coNP-complete, then the problem $\text{NAS}(Q)$ and hence the problem $\text{DAS}(Q)$ are in coNP, too (this is the reason why 1^s and not just s is part of the input of $\text{NAS}(Q)$ and of $\text{DAS}(Q)$). Thus, together with Lemma 4 the following lemma yields the remaining claims of Theorem 3.

- Lemma 5.** (a) Assume that $\text{DAS}(Q) \leq_p Q$, that is, that $\text{DAS}(Q)$ is polynomial time reducible to Q . If $\langle \mathbb{A}, 1^s \rangle \in \text{DAS}(Q)$ is not solvable in time $s^{f(\mathbb{A})}$ for some function f , then Q has hard sequences for algorithms.
- (b) Assume that $\text{NAS}(Q) \leq_p Q$. If there is no nondeterministic algorithm \mathbb{V} accepting $\text{NAS}(Q)$ such that for all $\langle \mathbb{A}, 1^s \rangle \in \text{NAS}(Q)$ we have $t_{\mathbb{V}}(\langle \mathbb{A}, 1^s \rangle) \leq s^{f(\mathbb{A})}$ for some function f , then Q has hard sequences for nondeterministic algorithms.

Proof. Again we only prove part (a).

Claim. Assume that $\langle \mathbb{A}, 1^s \rangle \in \text{DAS}(Q)$ is not solvable in time $s^{f(\mathbb{A})}$ for some function f . Then there is no algorithm \mathbb{W} deciding $\text{DAS}(Q)$ such that for all algorithms \mathbb{A} with $L(\mathbb{A}) \subseteq Q$ there is a $c_{\mathbb{A}} \in \mathbb{N}$ such that for all $s \in \mathbb{N}$ we have $t_{\mathbb{W}}(\langle \mathbb{A}, 1^s \rangle) \leq s^{c_{\mathbb{A}}}$.

Proof of the Claim. By contradiction, assume that such a \mathbb{W} exists. Let \mathbb{V} be the algorithm that, on an arbitrary input $\langle \mathbb{A}, 1^s \rangle$, in parallel runs \mathbb{W} on $\langle \mathbb{A}, 1^s \rangle$ and computes

$$r_{\mathbb{A}} := \text{the least } r \text{ such that } L(\mathbb{A}^r) \not\subseteq Q$$

by systematically checking for $r = 0, 1, \dots$ whether $L(\mathbb{A}^r) \not\subseteq Q$ (this is done by running for all x with $|x| \leq r$ the algorithm \mathbb{A} at most r steps on input x and a decision procedure for Q on x). Note that $r_{\mathbb{A}}$ is not defined if $L(\mathbb{A}) \subseteq Q$. If \mathbb{W} stops first, \mathbb{V} answers

accordingly; if $r_{\mathbb{A}}$ is obtained first, then \mathbb{V} accepts if $s < r_{\mathbb{A}}$ and otherwise it rejects. It should be clear that the algorithm \mathbb{V} decides $\langle \mathbb{A}, 1^s \rangle \in \text{DAS}(Q)$ in $\leq s^{f(\mathbb{A})}$ steps for some function f . \dashv

By assumption, there is a polynomial time reduction \mathbb{S} from $\text{DAS}(Q)$ to Q . Let \mathbb{B} be an arbitrary algorithm deciding Q . Then the algorithm $\mathbb{B} \circ \mathbb{S}$, which on input x first simulates \mathbb{S} on x and then \mathbb{B} on $\mathbb{S}(x)$, decides $\text{DAS}(Q)$. Hence, by the Claim, there exists an algorithm \mathbb{A} with $L(\mathbb{A}) \subseteq Q$ such that $t_{\mathbb{B} \circ \mathbb{S}}(\langle \mathbb{A}, 1^s \rangle)$ is not polynomially bounded in s . For $s \in \mathbb{N}$ we set $x_s := \mathbb{S}(\langle \mathbb{A}, 1^s \rangle)$. Then $x_s \in Q$ for all s and the function $1^s \mapsto x_s$ is polynomial time computable. Furthermore

$$t_{\mathbb{B} \circ \mathbb{S}}(\langle \mathbb{A}, 1^s \rangle) \leq O\left(t_{\mathbb{S}}(\langle \mathbb{A}, 1^s \rangle) + t_{\mathbb{B}}(\mathbb{S}(\langle \mathbb{A}, 1^s \rangle))\right) \leq s^{O(1)} + O(t_{\mathbb{B}}(x_s)).$$

As the left hand side is not polynomially bounded in s , neither is $t_{\mathbb{B}}(x_s)$. Hence $(x_s)_{s \in \mathbb{N}}$ is hard for \mathbb{B} . \square

Remark 6. Assume that Q is coNP-complete and has padding (the set TAUT is an example of such a Q). If Q has no almost optimal algorithm, then every algorithm \mathbb{B} deciding Q has a hard sequence $(x_s)_{s \in \mathbb{N}}$ with $s \leq |x_s|$. Then, in particular

$$\{x_s \mid s \in \mathbb{N}\} \in \text{P} \quad \text{and} \quad \mathbb{B} \text{ is not polynomial time on } \{x_s \mid s \in \mathbb{N}\}.$$

In fact, it is well-known that for Q with padding we can replace any polynomial time reduction to Q by a length-increasing one. Hence, in the previous proof we may assume that \mathbb{S} is length-increasing and therefore $s \leq |x_s|$.

Remark 7. In the proof of Theorem 3 we used the assumption that Q is coNP-complete only to ensure that $\text{NAS}(Q) \leq_p Q$ (cf. Lemma 5). This condition is also fulfilled for every Q complete, say, in one of the classes Π_t^p with $t \geq 1$, E or PSPACE. Thus the statements of Theorem 3 hold for such Q .

The argument in the last part of Lemma 5 shows (an instance of) the following simple lemma. Nevertheless, note that it is important that we do not require $s \leq |x_s|$ in our definition of hard sequence.

Lemma 8. *Assume that \mathbb{S} is a polynomial time reduction from Q to Q' and let \mathbb{B} be a (nondeterministic) algorithm deciding (accepting) Q' . If $(x_s)_{s \in \mathbb{N}}$ is a hard sequence for $\mathbb{B} \circ \mathbb{S}$, then $(\mathbb{S}(x_s))_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{B} .*

Therefore, if $Q \leq_p Q'$ and Q has hard sequences for (nondeterministic) algorithms then so does Q' .

We do not know proofs of the following results not using the machinery developed here.

Theorem 9. *Let Q be coNP-complete. Then, TAUT has an almost optimal algorithm if and only if Q has an almost optimal algorithm.*

Proof. Immediate by the previous lemma and Theorem 3. \square

We remark that the implication from left to right in the previous result was already known [7] (see also Theorem 12 below).

Theorem 10. *Assume that TAUT has no almost optimal algorithm. Then every coNP-hard problem has no almost optimal algorithm.*

Proof. By assumption and Theorem 3, TAUT has hard sequences for algorithms and so does every coNP-hard Q by Lemma 8. Now the claim follows from Lemma 2. \square

4. Hard sequences for proof systems

In this section we translate the results on hard sequences from algorithms to proof systems. We first recall some basic definitions.

A *proof system for Q* is a polynomial time algorithm \mathbb{P} computing a function from Σ^* onto Q . If $\mathbb{P}(w) = x$, we say that w is a \mathbb{P} -*proof* of x . Often we introduce proof systems implicitly by defining the corresponding function; then this definition will suggest a corresponding algorithm.

Definition 11. Let \mathbb{P} and \mathbb{P}' be proof systems for Q . An algorithm \mathbb{T} is a *translation from \mathbb{P}' into \mathbb{P}* if $\mathbb{P}(\mathbb{T}(w')) = \mathbb{P}'(w')$ for every $w' \in \Sigma^*$. Note that translations always exist. A translation is *polynomial* if it runs in polynomial time.

A proof system \mathbb{P} for Q is *p-optimal* or *polynomially optimal* if for every proof system \mathbb{P}' for Q there is a polynomial translation from \mathbb{P}' into \mathbb{P} . A proof system \mathbb{P} for Q is *optimal* if for every proof system \mathbb{P}' for Q and every $w' \in \Sigma^*$ there is a $w \in \Sigma^*$ such that $\mathbb{P}(w) = \mathbb{P}'(w')$ and $|w| \leq |w'|^{O(1)}$. Clearly, every p-optimal proof system is optimal.

We often will make use of the following relationship between the optimality notions for algorithms and that for proof systems (see [7, 11]).

Theorem 12. (1) *For every Q we have (a) \Rightarrow (b) and (b) \Rightarrow (c); moreover (a), (b), and (c) are all equivalent if Q has padding. Here*
 (a) *Q has a p-optimal proof system.*
 (b) *Q has an almost optimal algorithm.*
 (c) *There is an algorithm that decides Q and runs in polynomial time on every subset X of Q with $X \in \mathbf{P}$.*
 (2) *For every Q we have (a) \iff (b), (b) \Rightarrow (c), and (c) \Rightarrow (d); moreover (a)–(d) are all equivalent if Q has padding. Here*
 (a) *Q has an optimal proof system.*
 (b) *Q has an almost optimal nondeterministic algorithm.*
 (c) *There is a nondeterministic algorithm that accepts Q and runs in polynomial time on every subset X of Q with $X \in \mathbf{NP}$.*
 (d) *There is a nondeterministic algorithm that accepts Q and runs in polynomial time on every subset X of Q with $X \in \mathbf{P}$.*

We use our results of Section 3 to extend the equivalence between (a) and (b) of part (1) to arbitrary coNP-complete problems:

Theorem 13. *Let Q be coNP-complete. Then:*

Q has a p-optimal proof system \iff Q has an almost optimal algorithm.

Proof. By Theorem 12 (1) the left side implies the right side. Now assume that Q has an almost optimal algorithm. As $Q \times \Sigma^*$ is coNP-complete too, it has an almost optimal algorithm (by Theorem 9). As $Q \times \Sigma^*$ has padding, it has a p-optimal proof system \mathbb{P} (cf. Theorem 12 (1)). Now it is routine to show that the algorithm \mathbb{P}' that on input w computes $\mathbb{P}(w)$ and outputs its first component is a p-optimal proof system for Q . \square

We already mentioned that for every $Q \subseteq \Sigma^*$ there is a well-known and straightforward correspondence between proof systems and nondeterministic algorithms preserving the optimality notions, so that the proof of the equivalence between (a) and (b) in Theorem 12 (2) is immediate. In fact, if \mathbb{P} is a proof system for Q , then the nondeterministic algorithm $\mathbb{A}(\mathbb{P})$ accepts Q , where $\mathbb{A}(\mathbb{P})$ on input $x \in \Sigma^*$ guesses a string w and accepts if $\mathbb{P}(w) = x$. Conversely, if \mathbb{A} is a nondeterministic algorithm accepting Q , then for every fixed $x_0 \in Q$ a proof system $\mathbb{P}_{\mathbb{A}}$ for Q is defined by

$$\mathbb{P}_{\mathbb{A}}(w) := \begin{cases} x, & \text{if } w \text{ is a computation of } \mathbb{A} \text{ accepting } x \\ x_0, & \text{otherwise.} \end{cases}$$

The proof of the corresponding equivalence in Theorem 12 (1) is more involved and mostly, more or less explicitly, it is based on a theorem due to Levin on inverters. As we need this result, too, we recall it.

Let \mathbb{F} be an algorithm computing a function from Σ^* to Σ^* . An *inverter* of \mathbb{F} is an algorithm \mathbb{I} that given y in the range of \mathbb{F} halts with some output $\mathbb{I}(y)$ such that $\mathbb{F}(\mathbb{I}(y)) = y$. On inputs not in the range of \mathbb{F} , the algorithm \mathbb{I} may do whatever it wants. Levin [8] proved the following result.

Theorem 14. *Let \mathbb{F} be an algorithm computing a function from Σ^* into Σ^* . Then there is an optimal inverter that is, an inverter $\mathbb{O}_{\mathbb{F}}$ of \mathbb{F} such that for every inverter \mathbb{I} of \mathbb{F} and all y in the range of \mathbb{F} we have*

$$t_{\mathbb{O}_{\mathbb{F}}}(y) \leq (t_{\mathbb{I}}(y) + t_{\mathbb{F}}(\mathbb{I}(y)) + |y|)^{O(1)}.$$

Furthermore, $\mathbb{O}_{\mathbb{F}}$ does not halt on inputs y not in the range of \mathbb{F} .

We turn to hard sequences for proof systems.

Definition 15. Let \mathbb{P} be a proof systems for Q . A sequence $(x_s)_{s \in \mathbb{N}}$ is *hard (length-hard)* for \mathbb{P} if $\{x_s \mid s \in \mathbb{N}\} \subseteq Q$, the function $1^s \mapsto x_s$ is computable in polynomial time, and there is no polynomial time (nondeterministic) algorithm \mathbb{W} with $\mathbb{P}(\mathbb{W}(1^s)) = x_s$ for all $s \in \mathbb{N}$.

For nondeterministic \mathbb{W} by the unusual notation $\mathbb{P}(\mathbb{W}(1^s)) = x_s$ we mean that for every run of \mathbb{W} on 1^s outputting a string w we have $\mathbb{P}(w) = x_s$ and that there is at least one run that outputs a string. In more conventional terms, instead of “there is no polynomial time nondeterministic algorithm \mathbb{W} with $\mathbb{P}(\mathbb{W}(1^s)) = x_s$,” we equivalently could require that the function mapping 1^s to the minimum length in unary of a \mathbb{P} -proof of x_s is not polynomially bounded.

Definition 16. The problem Q has *hard (length-hard) sequences for proof systems* if every proof system for Q has a hard (length-hard) sequence.

As already remarked in the Introduction part (b) of the following result is due to Krajíček [6] who proved it by quite different means. Part (a) is already known for $Q = \text{TAUT}$ (see e.g. the survey [1, Section 11]). We give a new proof that works for any, not necessarily paddable coNP-complete problem Q .

Theorem 17. *Let Q be a coNP-complete problem. Then:*

- (a) *Q has no p -optimal proof system iff Q has hard sequences for proof systems.*
- (b) *Q has no optimal proof system iff Q has a length-hard sequence for proof systems.*

Proof. First we present a proof of the directions from right to left. Let \mathbb{P} be any proof system for Q . By our assumption on Q there is a hard (length-hard) sequence $(x_s)_{s \in \mathbb{N}}$ for \mathbb{P} . We consider the proof system \mathbb{P}' for Q by

$$\mathbb{P}'(w') := \mathbb{P}(w), \text{ if } w' = 0w; \quad \mathbb{P}'(w') := x_s, \text{ if } w' = 1^s;$$

and $\mathbb{P}'(w') := z_0$ for some fixed element z_0 of Q otherwise. By hardness (length-hardness) no translation from \mathbb{P}' into \mathbb{P} is polynomial (polynomially bounded): In fact, assume that $(x_s)_{s \in \mathbb{N}}$ is, say, length-hard for \mathbb{P} and by contradiction that the translation \mathbb{T} from \mathbb{P}' into \mathbb{P} is polynomially bounded. Let q be a polynomial such that $|\mathbb{T}(w')| \leq q(|w'|)$ for all w' . Then, the nondeterministic algorithm \mathbb{W} that on input 1^s guesses a string w of length $\leq q(s)$ and outputs it in case $\mathbb{P}(w) = x_s$ runs in polynomial time.

Now we present a proof of the direction from left to right; we do that only for (a) as that for (b) follows immediately from the result for algorithms by the simple correspondence between proof systems and nondeterministic algorithms mentioned above. So, assume that Q has no p -optimal proof system. By Theorem 13, Q has no almost optimal algorithm and hence has hard sequences for algorithms by Theorem 3.

Let \mathbb{P} be any proof system for Q . By Theorem 14, we have an inverter $\mathbb{O}_{\mathbb{P}}$ of \mathbb{P} which is optimal, that is, for every inverter \mathbb{I} of \mathbb{P} and $x \in Q$ we have

$$t_{\mathbb{O}_{\mathbb{P}}}(x) \leq (t_{\mathbb{I}}(x) + t_{\mathbb{P}}(\mathbb{I}(x)) + |x|)^{O(1)} \leq (t_{\mathbb{I}}(x) + |x|)^{O(1)}, \quad (2)$$

where the second inequality holds as $t_{\mathbb{P}}(w) \leq |w|^{O(1)}$ and hence $t_{\mathbb{P}}(\mathbb{I}(x)) \leq |\mathbb{I}(x)|^{O(1)} \leq t_{\mathbb{I}}(x)^{O(1)}$. Moreover, for $x \notin Q$ the algorithm $\mathbb{O}_{\mathbb{P}}$ will not halt on input x .

We choose an arbitrary algorithm \mathbb{Q} that decides Q and consider the algorithm \mathbb{S} that on input x in parallel simulates \mathbb{Q} and $\mathbb{O}_{\mathbb{P}}$, both on input x . If \mathbb{Q} halts first, then it answers accordingly and if $\mathbb{O}_{\mathbb{P}}$ halts first, then it accepts. Obviously \mathbb{S} decides Q and for every $x \in Q$ we have

$$t_{\mathbb{S}}(x) \leq O(t_{\mathbb{O}_{\mathbb{P}}}(x)). \quad (3)$$

As Q has hard sequences for algorithms, there is a polynomial time computable algorithm \mathbb{G} generating a hard sequence for \mathbb{S} , that is, \mathbb{G} on input 1^s computes $x_s \in Q$ in polynomial time such that

$$t_{\mathbb{S}}(x_s) \text{ is not polynomially bounded in } s. \quad (4)$$

Let \mathbb{G}^+ be the variant of the algorithm \mathbb{G}^* in the proof of Lemma 2 obtained by replacing Line 4 by

if this simulation outputs y and $y = x$ **then** output 1^s and halt.

Of course, on input $x = x_s$ the algorithm \mathbb{G}^+ runs in time polynomial in s . We show that $(x_s)_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{P} . So by contradiction, assume that \mathbb{W} is a polynomial time algorithm with $\mathbb{P}(\mathbb{W}(1^s)) = x_s$ for all $s \in \mathbb{N}$. We consider the inverter \mathbb{I} of \mathbb{P} that on input x in parallel simulates $\mathbb{O}_{\mathbb{P}}$ and \mathbb{G}^+ , both on input x . If $\mathbb{O}_{\mathbb{P}}$ halts, then it outputs the output of $\mathbb{O}_{\mathbb{P}}$ and halts; if \mathbb{G}^+ halts, then it simulates \mathbb{W} on $\mathbb{G}^+(x)$, outputs $\mathbb{W}(\mathbb{G}^+(x))$, and halts.

By definition of \mathbb{G}^+ the algorithm \mathbb{I} runs on input x_s in time polynomial in s , hence so does $\mathbb{O}_{\mathbb{P}}$ by (2) as $|x_s| \leq s^{O(1)}$. But then by (3), the same holds for the algorithm \mathbb{S} contradicting (4). \square

In the previous proof the hard (length-hard) sequence $(x_s)_{s \in \mathbb{N}}$ constructed for a proof system for Q was the hard sequence of a suitable (nondeterministic) algorithm for Q . Hence, by Remark 6, for Q with padding, we can require in Theorem 17 that for the claimed hard sequence $(x_s)_{s \in \mathbb{N}}$ we have $s \leq |x_s|$.

5. Hard subsets

As already remarked in the Introduction, if for an algorithm \mathbb{A} deciding a problem Q we have a hard sequence $(x_s)_{s \in \mathbb{N}}$ satisfying $s \leq |x_s|$, then $\{x_s \mid s \in \mathbb{N}\}$ is a polynomial time decidable subset of Q on which \mathbb{A} is not polynomial time. We then speak of a hard set for \mathbb{A} even if its elements cannot be generated in polynomial time. More precisely:

Definition 18. Let $Q \subseteq \Sigma^*$.

- (1) Let \mathbb{A} be a deterministic or nondeterministic algorithm accepting Q . A subset X of Q is *hard for \mathbb{A}* if $X \in \mathbb{P}$ and \mathbb{A} is not polynomial time on X .
- (2) The problem Q *has hard sets for algorithms* if every algorithm deciding Q has a hard set.
- (3) The problem Q *has hard sets for nondeterministic algorithms* if every nondeterministic algorithm accepting Q has a hard set.

Using these notions the equivalences (a) \Leftrightarrow (c) in Theorem 12 can be expressed in the following way:

Assume that Q has padding. Then

- (1) *Q has no almost optimal algorithm $\Leftrightarrow Q$ has hard sets for algorithms.*
- (2) *Q has no almost optimal nondeterministic algorithm $\Leftrightarrow Q$ has hard sets for nondeterministic algorithms.*

Hence, we get (we leave the nondeterministic variant to the reader):

Corollary 19. *Assume Q has padding.*

- (a) *If Q has hard sequences for algorithms, then Q has hard sets for algorithms.*
- (b) *If in addition Q is coNP-complete, then*

$$Q \text{ has hard sequences for algorithms} \Leftrightarrow Q \text{ has hard sets for algorithms.}$$

Proof. (a) If Q has hard sequences for algorithms, then, by Lemma 2, Q has no almost optimal algorithm and thus, by the previous remark, Q has hard sets for algorithms.

Again the previous remark together with Theorem 3 yields (b). \square

Assume that Q has an almost optimal algorithm. Then, in general, one cannot show that every algorithm deciding Q , which is not almost optimal, has a hard set. In fact, Messner [11, Corollary 3.33] has presented a P-immune Q_0 with an almost optimal algorithm. Of course, no algorithm deciding Q_0 has a hard set.

For an arbitrary problem Q the existence of hard subsets is equivalent to a (non-)listing property. We introduce this property.

Let C be the complexity class P or NP. A set X is a C -subset of Q if $X \subseteq Q$ and $X \in C$. Let C' be also one of the classes P or NP. We write $\text{List}(C, Q, C')$ and say that there is a *listing of the C -subsets of Q by C' -machines* if there is an algorithm that, once having been started, lists Turing machines $\mathbb{M}_1, \mathbb{M}_2, \dots$ of type C' such that

$$\{L(\mathbb{M}_i) \mid i \geq 1\} = \{X \subseteq Q \mid X \in C\}.$$

For Q with padding the equivalences in the following proposition were known [12].

Proposition 20. (1) Q has hard sets for algorithms \iff not $\text{List}(P, Q, P)$.
(2) Every nondeterministic algorithm \mathbb{A} accepting Q is not polynomial on at least one subset X of Q with $X \in \text{NP}$ \iff not $\text{List}(\text{NP}, Q, \text{NP})$.

Proof. We only prove the first claim as the second one can be obtained along the same lines. First we assume that not $\text{List}(P, Q, P)$. Let \mathbb{A} be an algorithm deciding Q . For $d \in \mathbb{N}$, by $\mathbb{A}(d)$ we denote the algorithm that on input x simulates \mathbb{A} on input x but rejects if the simulation exceeds time $|x|^d$.

We show that there is a P-subset X of Q such that

$$\text{for all } d: \quad X \not\subseteq \mathbb{A}(d)$$

Of course, then this X is hard for \mathbb{A} .

Otherwise, we fix an effective enumeration $\mathbb{D}_1, \mathbb{D}_2, \dots$ of all polynomial time Turing machines. Then $(\mathbb{D}_i(\mathbb{A}(j)))_{i,j \geq 1}$ is a listing of the P-subsets of Q , where $\mathbb{D}_i(\mathbb{A}(j))$ on input x , first simulates $\mathbb{A}(j)$ on x and if this algorithm accepts, then it simulates \mathbb{D}_i on input x and answers accordingly. In fact, as $\mathbb{A}(j)$ has to accept x , we have $L(\mathbb{D}_i(\mathbb{A}(j))) \subseteq Q$. And if X is a P-subset of Q accepted by \mathbb{D}_i , we choose a d such that $X \subseteq \mathbb{A}(d)$. Then $L(\mathbb{D}_i(\mathbb{A}(d))) = X$.

Conversely, assume that Q has hard sets for algorithms. By contradiction assume that \mathbb{L} is a listing witnessing $\text{List}(P, Q, P)$. Let \mathbb{Q} be an algorithm deciding Q . Consider the algorithm \mathbb{A} that on input x simulates \mathbb{Q} on x and in parallel for $i = 1, 2, \dots$ does the following:

- performs the i th step of \mathbb{L} ;
- if $\mathbb{M}_1, \dots, \mathbb{M}_s$ are the machines listed by \mathbb{L} so far, it performs an additional step of each of the \mathbb{M}_j s on x ; if one of these accepts it accepts.

If \mathbb{Q} halts first, it answers accordingly.

It should be clear that \mathbb{A} accepts Q . By assumption, there is a set X hard for \mathbb{A} . Let \mathbb{M}_{i_0} accept X . By definition of \mathbb{A} it should be clear that \mathbb{A} is polynomial on X , a contradiction. \square

We close this section by introducing hard subsets for proof systems and stating the corresponding result.

- Definition 21.** (1) Let \mathbb{P} be a proof system for Q . A subset X of Q is *hard (length-hard) for \mathbb{P}* if $X \in \mathbb{P}$ and there is no polynomial time (nondeterministic) algorithm \mathbb{W} such that $\mathbb{P}(\mathbb{W}(x)) = x$ for all $x \in X$ (cf. the remark after Definition 15 for the precise meaning of this last condition in the nondeterministic case).
- (2) Q has *hard (length-hard) sets for proof systems* if every proof system for Q has a hard (length-hard) set.

The following result can be obtained along the lines of the proof of Theorem 17. Again, due to the close relationship between nondeterministic algorithms and proof systems, part (b) can be viewed as a reformulation of the result for algorithms.

Theorem 22. *Let Q be a problem with padding. Then:*

- (a) Q has no p -optimal proof system if and only if Q has hard sets for proof systems.
(b) Q has no optimal proof system if and only if Q has length-hard sets for proof systems.

6. Assuming the Measure Hypothesis

In this section we present some examples of problems with special properties, some yield limitations to possible extensions of results mentioned in this paper. Most are proven assuming the Measure Hypothesis.

6.1. Complex sets with optimal algorithms and with optimal proof systems. For every $Q \in \text{NP}$, say, accepted by the polynomial time nondeterministic algorithm \mathbb{A} , the proof system \mathbb{P} is optimal, where $\mathbb{P}(w) := x$ if w is an accepting computation of \mathbb{A} on input x ; and otherwise, $\mathbb{P}(w) := z_0$ for some fixed element z_0 of Q . The question whether there are sets outside of NP with optimal proof systems was stated by Krajíček and Pudlák [7] and is still open. As already mentioned they proved that TAUT has an optimal proof system if $\text{E} = \text{NE}$.

We prove that there are problems in NE and outside of NP with optimal proof systems if the Measure Hypothesis holds. As a byproduct we get that there exist problems in E and outside of P with optimal algorithms (thereby we do not need the Measure Hypothesis). Here an algorithm \mathbb{A} deciding Q is *optimal* if for every algorithm \mathbb{B} deciding Q we have

$$t_{\mathbb{A}}(x) \leq (t_{\mathbb{B}}(x) + |x|)^{O(1)}$$

for all $x \in \Sigma^*$. Clearly, every problem in P has an optimal algorithm.

Let C be a class of problems. Recall that a problem Q is *C-immune* if no infinite subset of Q is in C; and it is *C-bi-immune* if Q and its complement $\Sigma^* \setminus Q$ are C-immune. For a function $t : \mathbb{N} \rightarrow \mathbb{N}$ we denote by $\text{DTIME}_0(t)$ and $\text{DTIME}(t)$ the class

of problems decidable by a Turing machine \mathbb{M} with $t_{\mathbb{M}}(x) \leq t(x)$ for all $x \in \Sigma^*$ and $t_{\mathbb{M}}(x) \leq c \cdot t(x)$ for all $x \in \Sigma^*$ and some constant $c \in \mathbb{N}$. The nondeterministic classes $\text{NTIME}_0(t)$ and $\text{NTIME}(t)$ are defined accordingly. Hence $\text{E} = \bigcup_{d \in \mathbb{N}} \text{DTIME}(2^{d \cdot n})$ and $\text{NE} = \bigcup_{d \in \mathbb{N}} \text{NTIME}(2^{d \cdot n})$.

Lemma 23. *Let $\ell \in \mathbb{N}$ with $\ell \geq 1$.*

- (a) *If $Q \in \text{E}$ is a $\text{DTIME}_0(2^{\ell \cdot n})$ -bi-immune problem, then Q has an optimal algorithm.*
- (b) *If $Q \in \text{NE}$ is a $\text{NTIME}_0(2^{\ell \cdot n})$ -immune problem, then Q has an almost optimal nondeterministic algorithm.*

Proof. We prove (a); part (b) is obtained by the obvious modifications. Assume that the Turing machine \mathbb{M} decides the $\text{DTIME}_0(2^{\ell \cdot n})$ -bi-immune problem Q in time $c \cdot 2^{d \cdot n}$ for some $c, d \in \mathbb{N}$. We claim that \mathbb{M} is optimal.

Assume otherwise, then there is a machine \mathbb{M}' deciding Q and witnessing that \mathbb{M} is not optimal. Then for every $i \in \mathbb{N}$ there exists an x_i such that

$$t_{\mathbb{M}}(x_i) > (t_{\mathbb{M}'}(x_i) + |x_i|)^i.$$

It follows that for every $i \in \mathbb{N}$

$$c \cdot 2^{d \cdot |x_i|} \geq t_{\mathbb{M}}(x_i) > t_{\mathbb{M}'}(x_i)^i$$

Thus $t_{\mathbb{M}'}(x_i) \leq 2^{\ell \cdot |x_i|/2}$ for all sufficiently large $i \in \mathbb{N}$. Of course, infinitely many of these x_i 's are in Q , or they are in $\Sigma^* \setminus Q$. In the first case consider the following machine:

\mathbb{M}'' // $x \in \Sigma^*$

1. simulate \mathbb{M}' on x for at most $2^{\ell \cdot |x|/2}$ steps
2. **if** the simulation halts and accepts **then** accept **else** reject.

It accepts an infinite subset of Q in time $2^{\ell \cdot n}$. This contradicts our immunity assumption. The second case is handled similarly. \square

We use the following result due to Mayordomo [10]. Statement (b) of it uses the *Measure Hypothesis* [5], that is, the assumption

NP does not have measure 0 in E.

For the corresponding notion of measure we refer to [10]. This hypothesis is sometimes used in the theory of resource bounded measures.

Theorem 24. *Let $\ell \geq 1$.*

- (a) *The class of $\text{DTIME}_0(2^{\ell \cdot n})$ -bi-immune problems has measure 1 in E. In particular, the class E contains $\text{DTIME}_0(2^{\ell \cdot n})$ -bi-immune problems.*
- (b) *If the Measure Hypothesis holds, then $\text{NP} \cap \text{E}$ contains $\text{DTIME}_0(2^{\ell \cdot n})$ -bi-immune problems.*

From the previous lemma and theorem we get:

- Corollary 25.** (1) *There exist problems in $E \setminus P$ with optimal algorithms.*
(2) *If the Measure Hypothesis holds, then there exist problems in $NP \setminus P$ with optimal algorithms.*

We already remarked that Messner [11] showed the existence of problems in $E \setminus P$ with almost optimal algorithms.

Theorem 26. *If the Measure Hypothesis holds, then there exist problems in $NE \setminus NP$ with optimal proof systems.*

Proof. It suffices to show that there is a $Q \in NE$ which is $\text{NTIME}_0(2^n)$ -immune. Then, by Lemma 23, such a Q has an almost optimal nondeterministic algorithm and hence, an optimal proof system by Theorem 12.

By Theorem 24 (b) there is a $Q_0 \in NP$ which is $\text{DTIME}_0(2^{2n})$ -bi-immune problem. We choose $d \geq 1$ such that $Q_0 \in \text{NTIME}(n^d)$. We set

$$Q := \{1^m \mid m \in \mathbb{N} \text{ and } 1^{2^m} \in Q_0\}.$$

Then $Q \in NE$. Furthermore, Q is infinite as otherwise the set $\{1^{2^m} \mid m \in \mathbb{N} \text{ and } 1^{2^m} \notin Q_0\}$ would be an infinite subset of $\Sigma^* \setminus Q_0$ in P contradicting the bi-immunity property of Q_0 . Finally we show that Q is $\text{NTIME}_0(2^n)$ -immune. By contradiction assume that there is an infinite $S \subseteq Q$ accepted by a nondeterministic algorithm \mathbb{S} in time 2^n . Then the set

$$S^* := \{1^n \mid n = 2^m \text{ for some } m \in \mathbb{N} \text{ and } 1^m \in S\}$$

is an infinite subset of Q_0 . The algorithm that first computes m from 1^n and then deterministically simulates all possible runs of \mathbb{S} on 1^m runs in time

$$n^{O(1)} + O(2^{2^m}) = n^{O(1)} + O(2^n) \leq 2^{2n}$$

for sufficiently large n . This contradicts the $\text{DTIME}_0(2^{2n})$ -immunity of Q_0 . \square

6.2. Non-optimal algorithms without hard sequences. In this final part we show that, assuming the Measure Hypothesis,

- every problem with padding and with an almost optimal algorithm has an algorithm which is not almost optimal but has no hard sequence
- there is a problem without almost optimal algorithm having an algorithm without hard sequence.

Our proofs are based on the following proposition.

Proposition 27. *If the Measure Hypothesis holds, then there is a problem $Q_0 \in P$ such that*

- (a) *there is an algorithm \mathbb{B} deciding Q_0 which is not almost optimal (or, equivalently, is not polynomial time) but has no hard sequences;*
- (b) *every algorithm \mathbb{A} deciding Q_0 with*

$$t_{\mathbb{A}}(x) \leq 2^{e \cdot (\log |x|)^2}$$

for every $x \in \Sigma^$ and some constant $e \geq 1$ has no hard sequences;*

(c) there is a proof system for Q_0 which is not optimal but has no hard sequences.

In the proof we shall use:

Lemma 28. Let \mathbb{A} be an algorithm deciding a problem Q_0 with

$$t_{\mathbb{A}}(x) \leq 2^{e \cdot (\log |x|)^2} \quad (5)$$

for all $x \in \Sigma^*$ and some $e \geq 1$. Assume that $(x_s)_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{A} . Then there is a sequence $s_0 < s_1 < s_2 < \dots$ such that

$$\lim_{i \rightarrow \infty} \frac{\log s_i}{(\log |x_{s_i}|)^2} = 0 \quad \text{i.e.,} \quad s_i = 2^{o((\log |x_{s_i}|)^2)}.$$

In particular, the set $\{x_{s_i} \mid i \in \mathbb{N}\}$ is infinite.

Proof. Assume otherwise that for some $\varepsilon > 0$ and some $n \in \mathbb{N}$ and all $s \geq n$

$$\frac{\log s}{(\log |x_s|)^2} \geq \varepsilon,$$

or equivalently, $s \geq 2^{\varepsilon \cdot (\log |x_s|)^2}$; then $s \geq t_{\mathbb{A}}(x_s)^{\varepsilon/e}$ by assumption. This contradicts the hardness of $(x_s)_{s \in \mathbb{N}}$. \square

Proof of Proposition 27. (a) and (b): By the Measure Hypothesis there is a $\text{DTIME}_0(2^n)$ -bi-immune $Q_1 \in \text{NP}$. In particular, there exists a nondeterministic Turing machine \mathbb{M} with binary nondeterminism and a $d \in \mathbb{N}$ such that for all $y \in \Sigma^*$ (with $|y| \geq 2$) the machine \mathbb{M} decides whether $y \in Q_1$ in $\leq |y|^d$ steps. Thus for $y \in \Sigma^*$ every string $x \in \{0, 1\}^{|y|^d}$ determines a unique run of \mathbb{M} on y . We set

$$Q_0 := \left\{ x \in \{0, 1\}^* \mid \text{for some } n \in \mathbb{N} \text{ we have } |x| = n^d \right. \\ \left. \text{and } x \text{ determines an accepting run of } \mathbb{M} \text{ on input } 1^n \right\}.$$

Then Q_0 is infinite, as otherwise the set $\{1^n \in Q_1 \mid n \in \mathbb{N}\}$ would be finite contradicting the $\text{DTIME}_0(2^n)$ -bi-immunity of Q_1 . Clearly $Q_0 \in \text{P}$. Let \mathbb{A}_0 be an algorithm deciding Q_0 in polynomial time and let \mathbb{B} be the algorithm deciding Q_0 by first simulating \mathbb{A} , and then making an appropriate number of dummy steps such that for some $e \geq 1$ and all $y \in \Sigma^*$

$$t_{\mathbb{B}}(y) = 2^{e \cdot (\log |y|)^2}. \quad (6)$$

Then \mathbb{A}_0 witnesses that \mathbb{B} is not almost optimal.

We finish our proof by showing that for every algorithm \mathbb{A} deciding Q_0 such that for some $e \geq 1$ and all $y \in \Sigma^*$

$$t_{\mathbb{A}}(y) \leq 2^{e \cdot (\log |y|)^2}.$$

has no hard sequences. Towards a contradiction assume \mathbb{A} has a hard sequence $(x_s)_{s \in \mathbb{N}}$. We set

$L_0 := \{1^n \mid \text{for some } s \in \mathbb{N}, |x_s| = n^d \text{ and } x_s \text{ determines an accepting run of } \mathbb{M} \text{ on } 1^n\}$.

Clearly, $L_0 \subseteq Q_1$. We choose a polynomial time algorithm \mathbb{G} computing the function $1^s \mapsto x_s$. The following algorithm \mathbb{C} accepts L_0 .

```

 $\mathbb{C}$     //  $y \in \Sigma^*$ 
1.  $n \leftarrow |y|$ 
2. if  $y \neq 1^n$  then reject
3.  $\ell \leftarrow 0$ 
4. for  $s = 0$  to  $\ell$ 
5.     simulate the  $(\ell - s)$ th step of  $\mathbb{G}$  on  $1^s$ 
6.     if the simulation outputs  $x$  with  $|x| = n^d$  then accept
7.  $\ell \leftarrow \ell + 1$ 
8. goto 3.

```

By (6) we can apply Lemma 28 to \mathbb{A} and get a sequence $s_0 < s_1 < s_2 < \dots$. For $i \in \mathbb{N}$ we let

$$n_i := \sqrt[d]{|x_{s_i}|}. \quad (7)$$

Hence, x_{s_i} is an accepting run of \mathbb{M} on input 1^{n_i} . We show that

$$t_{\mathbb{C}}(1^{n_i}) = 2^{o((\log n_i)^2)}. \quad (8)$$

In fact, as \mathbb{G} runs in polynomial time, we have $|x_{s_i}| \leq |s_i|^{O(1)}$, and by (7) therefore, $|n_i| \leq |s_i|^{O(1)}$. Now one easily sees that \mathbb{C} accepts 1^{n_i} in time polynomial in s_i , too. By Lemma 28

$$s_i = 2^{o((\log |x_{s_i}|)^2)}.$$

Thus (7) implies that

$$s_i = 2^{o((\log n_i)^2)}.$$

Hence, we get (8).

Finally, we consider the algorithm \mathbb{C}^* that on input y simulates \mathbb{C} for $2^{|y|}$ steps and accepts if the simulation accepts. By (8), \mathbb{C}^* accepts an infinite subset of L_0 . As $L_0 \subseteq Q_1$, this contradicts the $\text{DTIME}_0(2^n)$ -bi-immunity of Q_1 .

(c) Let Q_0 and \mathbb{B} be as in part (a). We leave it to the reader to show that the following proof system \mathbb{P} for Q_0 is not optimal but has no hard sequence. For $w \in \Sigma^*$ let

$$\mathbb{P}(w) := x, \quad \text{if } w \text{ is a computation of } \mathbb{B} \text{ accepting } x$$

and $\mathbb{P}(w) := z_0$ for some fixed $z_0 \in Q_0$ otherwise. \square

Theorem 29. *Let Q be a problem with padding and with an almost optimal algorithm. If the Measure Hypothesis holds, then there is an algorithm deciding Q , which is not optimal, has hard sets but does not have hard sequences.*

Proof. Let pad and \mathbb{O} be a padding function and an almost optimal algorithm for Q , respectively. With Proposition 27 (a) choose a $Q_0 \in \mathbf{P}$ and an algorithm \mathbb{B} deciding Q_0 which is not almost optimal but has no hard sequences. Fix $z_0 \in Q$ and let \mathbb{A} be the algorithm deciding Q that on input x first checks (in polynomial time) whether $x = pad(z_0, y)$ with $y \in Q_0$ (using the properties of the padding function and a polynomial time algorithm deciding Q_0); if so, it simulates \mathbb{B} on y ; otherwise it simulates \mathbb{O} on x .

Clearly, \mathbb{A} is not almost optimal as it can be speeded up on the set $\{pad(z_0, y) \mid y \in Q_0\}$, a hard set of \mathbb{A} . By contradiction, assume $(x_s)_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{A} and let $y_0 \in Q_0$. For $s \geq 1$ we set

$$y_s := \begin{cases} y, & \text{if } x_s = pad(z_0, y) \text{ with } y \in Q_0 \\ y_{s-1}, & \text{otherwise} \end{cases}$$

and

$$z_s := \begin{cases} z_{s-1}, & \text{if } x_s = pad(z_0, y) \text{ with } y \in Q_0 \\ x_s, & \text{otherwise.} \end{cases}$$

Then either $(y_s)_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{B} or $(z_s)_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{O} , in both cases a contradiction. \square

Corollary 30. *If the Measure Hypothesis holds, then the following are equivalent:*

- (i) *Every coNP-complete problem has no almost optimal algorithm.*
- (ii) *Every non-almost optimal algorithm deciding a coNP-complete problem has hard sequences.*

Proof. We already know that (i) implies (ii) by Theorem 3 (a). Assume (ii) and by contradiction, suppose that Q is a coNP-complete problem with an almost optimal algorithm. By Theorem 9, we may assume that Q has padding. Then, by the previous theorem, there is a non-almost optimal algorithm deciding Q without hard sequences, contradicting (ii). \square

The following example shows that the padding hypothesis is necessary in Theorem 29.

Example 1. Let $Q := \{1^n \mid n \in \mathbb{N}\}$. As $Q \in \mathbf{P}$, it has an almost algorithm. However, the set Q itself is a hard set and $(1^s)_{s \in \mathbb{N}}$ a hard sequence for every non-optimal (that is, for every superpolynomial) algorithm deciding Q .

Finally, we show that also problems *without* almost optimal algorithm may have algorithms without hard sequences:

Theorem 31. *If the Measure Hypothesis holds, there is a problem which has hard sets for algorithms (and hence has no almost optimal algorithm) but has algorithms without hard sequences.*

Proof. Let $Q_0 \in \mathbf{P}$ be a problem with the properties stated in Proposition 27. We fix an effective enumeration

$$\mathbb{A}_0, \mathbb{A}_1, \dots, \tag{9}$$

of all algorithms such that there is an universal algorithm \mathbb{U} which on every input $\langle 1^i, x \rangle$ simulates the algorithm \mathbb{A}_i on input $\langle 1^i, x \rangle$ in such a way that

$$t_{\mathbb{U}}(\langle 1^i, x \rangle) \leq (i + 1) \cdot t_{\mathbb{A}_i}(\langle i, x \rangle)^2. \quad (10)$$

For every $i \in \mathbb{N}$ we let

$$S_i := \left\{ \langle 1^i, x \rangle \mid x \in Q_0 \text{ and } \mathbb{A}_i \text{ does not accept } \langle 1^i, x \rangle \text{ in } \leq 2^{(\log |x|)^2} \text{ steps} \right\}. \quad (11)$$

Finally, we set

$$Q := \bigcup_{i \in \mathbb{N}} S_i.$$

and show that Q is a problem with the properties mentioned in the theorem.

Claim 1. Let $k \in \mathbb{N}$. If \mathbb{A}_k (see (9)) decides Q , then $S_k = \{\langle 1^k, x \rangle \mid x \in Q_0\}$.

Proof of Claim 1. Otherwise, there exists an $x_0 \in Q_0$ with $\langle 1^k, x_0 \rangle \notin S_k$. It follows that

$$\begin{aligned} x_0 \in Q_0 \text{ with } \langle 1^k, x_0 \rangle \notin S_k &\implies \mathbb{A}_k \text{ accepts } \langle 1^k, x_0 \rangle \text{ in } \leq 2^{(\log |x|)^2} \text{ steps} && \text{(by (11))} \\ &\implies \mathbb{A}_k \text{ accepts } \langle 1^k, x_0 \rangle \\ &\implies \langle 1^k, x_0 \rangle \in Q && \text{(as } \mathbb{A}_k \text{ decides } Q) \\ &\implies \langle 1^k, x_0 \rangle \in S_k && \text{(since all } S_i \text{'s are disjoint).} \end{aligned}$$

This is a contraction. \dashv

Claim 2. Q has hard sets for algorithms.

Proof of Claim 2. Assume that \mathbb{A}_k decides Q . By Claim 1, $S_k = \{\langle 1^k, x \rangle \mid x \in Q_0\}$ and by (11) for every $x \in Q_0$,

$$t_{\mathbb{A}_k}(\langle 1^k, x \rangle) > 2^{(\log |x|)^2}.$$

As $Q_0 \in \mathbf{P}$, thus S_k is a hard set for \mathbb{A}_k . \dashv

Claim 3. For all sufficiently large $d \in \mathbb{N}$ there is an algorithm \mathbb{Q}_d deciding Q such that

$$t_{\mathbb{Q}_d}(\langle 1^i, x \rangle) = (i + 1) \cdot 2^{d \cdot (\log |x|)^2}$$

for every $i \in \mathbb{N}$ and $x \in \Sigma^*$.

Proof of Claim 3. By (10) and (11) as $Q_0 \in \mathbf{P}$. \dashv

Now we choose a sufficiently large $d \in \mathbb{N}$ and consider the algorithm \mathbb{Q}_d of Claim 3. Assume that \mathbb{Q}_d has a hard sequence

$$\left(\langle 1^{i_s}, x_s \rangle \right)_{s \in \mathbb{N}}.$$

By (11) every x_s is in Q_0 and by hardness,

$$t_{\mathbb{Q}_d}(\langle 1^{i_s}, x_s \rangle) = (i_s + 1) \cdot 2^{d \cdot (\log |x_s|)^2}$$

is superpolynomial in s . Since the mapping $1^s \mapsto \langle 1^{i_s}, x_s \rangle$ is computable in polynomial time, we have $|i_s| \leq |s|^{O(1)}$. Therefore,

$$2^{d \cdot (\log |x_s|)^2} \text{ is superpolynomial in } s. \quad (12)$$

As Q_0 is decidable in polynomial time and d is sufficiently large, we have an algorithm \mathbb{A} deciding Q_0 in time $2^{d \cdot (\log |x|)^2}$ on every instance $x \in \Sigma^*$. Then (12) implies that $(x_s)_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{A} , which contradicts Proposition 27 (b). \square

7. Getting hard sequences in an effective way

We have mentioned in the Introduction that McCreight and Meyer [9] have shown that for every EXP-hard problem Q there is a polynomial time procedure assigning to every algorithm deciding Q a hard sequence. Based on their proof we derive a “nondeterministic” version.

Theorem 32. *Let Q be a coNEXP-hard problem. Then there is a polynomial time computable function $g : \Sigma^* \times \{1\}^* \rightarrow \Sigma^*$ such that for every nondeterministic algorithm \mathbb{A} accepting Q the sequence $(g(\mathbb{A}, 1^s))_{s \in \mathbb{N}}$ is hard for \mathbb{A} .*

Proof. Consider the problem

Q_0 <i>Instance:</i> A nondeterministic algorithm \mathbb{A} . <i>Question:</i> Is it true that \mathbb{A} does not accept \mathbb{A} in at most $2^{ \mathbb{A} }$ steps?
--

Claim 1. If \mathbb{B} is a nondeterministic algorithm accepting Q_0 , then $\mathbb{B} \in Q_0$ and therefore, $t_{\mathbb{B}}(\mathbb{B}) > 2^{|\mathbb{B}|}$.

Proof of Claim 1. Assume that $\mathbb{B} \notin Q_0$. Therefore, \mathbb{B} does not accept \mathbb{B} . Then, by the definition of Q_0 , we have $\mathbb{B} \in Q_0$, a contradiction. \dashv

To every nondeterministic algorithm \mathbb{A} and every $s \in \mathbb{N}$ we can assign in time polynomial in \mathbb{A} and s a nondeterministic algorithm \mathbb{A}_s with

$$|\mathbb{A}_s| \geq s, \quad L(\mathbb{A}_s) = L(\mathbb{A}), \quad \text{and} \quad t_{\mathbb{A}_s} = t_{\mathbb{A}} \quad (13)$$

(say, by adding s new “dummy” states).

Claim 2. If \mathbb{A} is a nondeterministic algorithm accepting Q_0 , then $(\mathbb{A}_s)_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{A} .

Proof of Claim 2. It suffices to verify for all $s \in \mathbb{N}$

$$\mathbb{A}_s \in Q_0 \quad (14)$$

$$t_{\mathbb{A}}(\mathbb{A}_s) > 2^s. \quad (15)$$

By (13) we know that $L(\mathbb{A}_s) = L(\mathbb{A})$. Hence, (14) holds by Claim 1, which also shows the first inequality in

$$t_{\mathbb{A}}(\mathbb{A}_s) = t_{\mathbb{A}_s}(\mathbb{A}_s) > 2^{|\mathbb{A}_s|} \geq 2^s,$$

the second one and the equality holding by (13). \dashv

Now let Q be coNEXP-hard. Since $Q_0 \in \text{coNEXP}$ there is a polynomial time reduction \mathbb{S} from Q_0 to Q . Again, for a nondeterministic algorithm \mathbb{A} let $\mathbb{A} \circ \mathbb{S}$ be the nondeterministic algorithm that on input $x \in \Sigma^*$ first runs \mathbb{S} on x and then runs \mathbb{A} on $\mathbb{S}(x)$.

For a nondeterministic algorithm \mathbb{A} and $s \in \mathbb{N}$ we define

$$g(\mathbb{A}, 1^s) := \mathbb{S}((\mathbb{A} \circ \mathbb{S})_s).$$

Clearly, g is polynomial time computable. If \mathbb{A} decides Q , then $\mathbb{A} \circ \mathbb{S}$ decides Q_0 ; therefore, $((\mathbb{A} \circ \mathbb{S})_s)_{s \in \mathbb{N}}$ is a hard sequence for $\mathbb{A} \circ \mathbb{S}$ by Claim 2. Hence, $(g(\mathbb{A}, 1^s))_{s \in \mathbb{N}}$ is a hard sequence for \mathbb{A} by Lemma 8. \square

Acknowledgements. The authors wish to thank the John Templeton Foundation for its support under Grant #13152, *The Myriad Aspects of Infinity*. This research also has been partially supported by the National Nature Science Foundation of China (60970011), the Sino-German Center for Research Promotion (GZ584). Yijia Chen is affiliated with BASICS and MOE-MS Key Laboratory for Intelligent Computing and Intelligent Systems which is supported by National Nature Science Foundation of China (61033002).

References

1. O. Beyersdorff. On the correspondence between arithmetic theories and propositional proof systems - a survey. *Mathematical Logic Quarterly*, 55(2):116–137, 2009.
2. Y. Chen and J. Flum. On p-optimal proof systems and logics for PTIME. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP'10, Track B)*, volume 6199 of *Lecture Notes in Computer Science*, pp. 321–322, 2010.
3. Y. Chen and J. Flum. Listings and logics. Electronic Colloquium on Computational Complexity (ECCC), TR11-020, 2011.
4. S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44:36–50, 1979.
5. J.M. Hitchcock and A. Pavan. Hardness hypotheses, derandomization, and circuit complexity. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, 336–347, 2004.
6. J. Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*. Cambridge University Press, 1995.
7. J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54:1063–1088, 1989.
8. L. Levin. Universal search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.
9. A. Meyer. A supervisor's reminiscence what we were thinking. Talk at the Stockmeyer-Symposium, 2005.
10. E. Mayordomo. Almost every set in exponential time is P-bi-immune. *Theoretical Computer Science*, 136(2): 487–506, 1994.

11. J. Messner. On the simulation order of proof systems. PhD Thesis, University of Erlangen, 2000.
12. Z. Sadowski. On an optimal propositional proof system and the structure of easy subsets of TAUT. *Theoretical Computer Science*, 288(1):181–193, 2002.
13. L. Stockmeyer. The Complexity of Decision Problems in Automata Theory. *PhD. Thesis*, MIT 1974.