Hard Instances of Algorithms and Proof Systems

Yijia Chen[1], Jörg Flum[2], and Moritz Müller[3]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Dongchuan Road, No. 800, 200240 Shanghai, China
yijia.chen@cs.sjtu.edu.cn

[2] Abteilung für mathematische Logik, Albert-Ludwigs-Universität Freiburg, Eckerstraße 1, 79104 Freiburg, Germany
joerg.flum@math.uni-freiburg.de

[3] Kurt Gödel Research Center for Mathematical Logic, Währinger Straße 25, 1090 Wien, Austria
moritz.mueller@univie.ac.at

Abstract. If the class Taut of tautologies of propositional logic has no almost optimal algorithm, then every algorithm $\mathbb{A}$ deciding Taut has a polynomial time computable sequence witnessing that $\mathbb{A}$ is not almost optimal. We show that this result extends to every Π_t^p-complete problem with $t \geq 1$; however, assuming the Measure Hypothesis, there is a problem which has no almost optimal algorithm but is decided by an algorithm without such a hard sequence. Assuming that a problem Q has an almost optimal algorithm, we analyze whether every algorithm deciding Q, which is not almost optimal algorithm, has a hard sequence.

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.[1] 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 any other algorithm $\mathbb{B}$ deciding Q and all $x \in Q$ the running time $t_{\mathbb{A}}(x)$ is polynomially bounded in $t_{\mathbb{B}}(x)$. In fact, if $(x_s)_{s\in\mathbb{N}}$ is a hard sequence for an algorithm, then one can superpolynomially 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 starting point is the following result (more or less explicit in [3,11]):

If Taut*, the class of tautologies of propositional logic, has no almost optimal algorithm, then every algorithm deciding* Taut *has hard sequences.*

First we generalize this result from the Π_1^p-complete problem Taut to all problems which are Π_t^p-complete for some $t \geq 1$:

[1] All notions will be defined in a precise manner later.

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

Apparently there are some limitations when trying to show (∗) for all problems Q as we prove:

(+) *If the Measure Hypothesis holds, then there is a problem which has no almost optimal algorithm but is decided by an algorithm without hard sequences.*

Perhaps one would expect that one can strengthen (∗) and show that even if a Π_t^p-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 problem with padding and with an almost optimal algorithm is decided by an algorithm which is not almost optimal but has no hard sequences.

As an algorithm deciding a problem Q which is not almost optimal can be polynomially speeded up on an infinite subset of Q, by (+) we see that, at least under the Measure Hypothesis, this notion of speeding up (e.g., considered in [13]) is weaker than our notion of the existence of a hard sequence.

Assume that $Q :=$ Taut (or any Π_t^p-complete Q) has no almost optimal algorithm; thus, by (∗), every algorithm deciding Q has a hard sequence. 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 Stockmeyer [13] and rediscovered by Messner [10] 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 EXP $= \Pi_t^p$, then for every Π_t^p-hard problem Q there is a polynomial time effective procedure assigning a hard sequence to every algorithm deciding Q.

Our proof of (∗) generalizes to nondeterministic algorithms. This "nondeterministic statement" yields a version for Π_t^p-complete problems of a result that Krajíček derived for non-optimal propositional proof systems: If Taut 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 which only have superpolynomial $\mathbb{P}$-proofs; moreover, he showed that the α_s can be chosen with $s \leq |\alpha_s|$. While it is well-known that for any problem Q nondeterministic algorithms deciding Q and proof systems for Q are more or less the same, the relationship between deterministic algorithms and propositional proof systems is more subtle. Nevertheless, we are able to use (∗) to derive a statement on hard sequences for Π_t^p-complete problems Q without a *polynomially* optimal proof system.

As a byproduct, we obtain results in "classical terms" (that is, not referring to hard sequences). For example, we get for $t \geq 1$:

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

If some Π_t^p-complete has no almost optimal algorithm, then every Π_t^p-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 NE = coNE implies that TAUT has an optimal proof system, a result later strengthened by [8,1].

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 [10] 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. [12,3,4]). 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. Due to space limitations we defer almost all proofs to the full version of this extended abstract.

2 Preliminaries

By $n^{O(1)}$ we denote the class of polynomially bounded functions on the natural numbers. We let Σ be the alphabet $\{0, 1\}$ and $|x|$ 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 assume familiarity with the classes P (polynomial time), NP (nondeterministic polynomial time) and the classes Π_t^p for $t \geq 1$ (the "universal" class of the tth level of the polynomial hierarchy). In particular, Π_1^p = coNP.

The *Measure Hypothesis* [5] is the assumption "NP does not have measure 0 in E."' For the corresponding notion of measure we refer to [9]. This hypothesis is sometimes used in the theory of resource bounded measures.

A problem $Q \subseteq \Sigma^*$ *has padding* if there is a function $pad : \Sigma^* \times \Sigma^* \to \Sigma^*$ computable in logarithmic space having the following properties:

– For any $x, y \in \Sigma^*$, $|pad(x, y)| > |x| + |y|$ and $\big(pad(x, y) \in Q \iff x \in Q\big)$.
– 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 shall 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 Π_t^p-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

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

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

Clearly, every problem in P (NP) has an almost optimal (nondeterministic) algorithm. There are problems outside P with an almost optimal algorithm (see Messner[10, Corollary 3.33]; we slightly improve his result in Theorem 22 of 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 Theorem 23 of Section 6 that the former is true if the Measure Hypothesis holds.

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* (*for nondeterministic algorithms*) if every (nondeterministic) algorithm deciding Q has a hard sequence.

In the proof of the following lemma we show that an algorithm $\mathbb{A}$ can be superpolynomially speeded up on $\{x_s \mid s \in \mathbb{N}\}$ if $(x_s)_{s\in\mathbb{N}}$ is hard for $\mathbb{A}$.

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.*

As already remarked in the Introduction, part (b) of the next theorem, the main result of this section, generalizes the corresponding result for Q = TAUT due to Krajíček.

Theorem 3. *Let Q be a Π^p_t-complete problem for some $t \geq 1$. 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.

Remark 4. For Q = Taut, part (a) is implicit in [3,11]. In fact, there it is shown that a halting problem polynomially isomorphic to Taut has hard sequences for algorithms if it has no almost optimal algorithm. In Remark 14 we show how this can be extended to every coNP-complete problem using known results relating almost optimal algorithms and proof systems.

Lemma 2 yields the implications from right to left in Theorem 3. 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.

For $Q \subseteq \Sigma^*$ we consider the *deterministic (nondeterministic) algorithm subset problem* Das(Q) (Nas(Q))

Das(Q) (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 Das(Q) (of Nas(Q)).

Lemma 5. *(a) If $\langle \mathbb{A}, 1^s \rangle \in$ 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 Nas(Q) *such that for all $\langle \mathbb{A}, 1^s \rangle \in$* 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.*

If Q is Π^p_t-complete, then Nas(Q) and hence Das(Q) are in Π^p_t, too (this is the reason why 1^s and not just s is part of the input of Nas(Q) and of Das(Q)). Thus, together with Lemma 5 the following lemma yields the remaining claims of Theorem 3.

Lemma 6. *(a) Assume that* Das(Q) $\leq_p Q$*, that is, that* Das(Q) *is polynomial time reducible to Q. If $\langle \mathbb{A}, 1^s \rangle \in$* 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* $\textsc{Nas}(Q) \leq_p Q$. *If there is no nondeterministic algorithm* $\mathbb{V}$ *accepting* $\textsc{Nas}(Q)$ *such that for all* $\langle \mathbb{A}, 1^s \rangle \in \textsc{Nas}(Q)$ *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.*

Remark 7. In the proof of Theorem 3 we use the assumption that Q is Π_t^p-complete only to ensure that $\textsc{Nas}(Q) \leq_p Q$ (cf. Lemma 6). This condition is also fulfilled for every Q complete, say, in one of the classes E or $\textsc{Pspace}$. Thus the statements of Theorem 3 hold for such a Q.

Remark 8. Assume that Q is Π_t^p-complete and has padding (for $t = 1$, the set $\textsc{Taut}$ 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

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

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. An analysis of the proof of Lemma 6 shows that then we can get hard sequences $(x_s)_{s\in\mathbb{N}}$ with $s \leq |x_s|$.

In contrast to the last remark, for the validity of the next lemma it is important that we do not require $s \leq |x_s|$ in our definition of hard sequence.

Lemma 9. *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 derive two consequences of our results:

Corollary 10. *Assume* $t \geq 1$ *and let* Q *and* Q' *be* Π_t^p*-complete. Then,* Q *has an almost optimal algorithm if and only if* Q' *has an almost optimal algorithm.*

Corollary 11. *Let* $t \geq 1$ *and assume that the some* Π_t^p*-complete problem has no almost optimal algorithm. Then every* Π_t^p*-hard problem has no almost optimal algorithm.*

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.

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 all $w' \in \Sigma^*$ there is a $w \in \Sigma^*$ such that $\mathbb{P}(w) = \mathbb{P}'(w')$ and $|w| \leq |w'|^{O(1)}$. Clearly, any 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 [7,10]).

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 $X \in \mathrm{P}$.
(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 $X \in \mathrm{NP}$.
(d) There is a nondeterministic algorithm that accepts Q and runs in polynomial time on every subset X of Q with $X \in \mathrm{P}$.

We use our results of Section 3 to extend the equivalence between (a) and (b) of part (1) of Theorem 12 to arbitrary Π^p_t-complete problems:

Theorem 13. *Let Q be a Π^p_t-complete problem for some $t \geq 1$. Then:*

$$Q \text{ has a p-optimal proof system} \iff Q \text{ has an almost optimal algorithm.}$$

Remark 14. Using Theorem 12, for every coNP-complete Q we get a simple, direct proof of

if Q has no almost optimal algorithm, then Q has hard sequences for algorithms

using the result for Q = Taut (that we already knew by Remark 4). In fact, assume that Q has no almost optimal algorithm. Then Taut has no almost optimal algorithm; otherwise, Taut has a p-optimal proof system by the equivalence of (a) and (b) in part (1) of Theorem 12 (Taut has padding!). As $Q \leq_p$ Taut, then Q has a p-optimal proof system too (cf. [8, Lemma 1]) and hence, again by Theorem 12, an almost optimal algorithm, a contradiction. Thus, Taut has hard sequences for algorithms. As Taut $\leq_p Q$, by Lemma 9 the problem Q has hard sequences for algorithms, too.

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. Thus the translation of our results for nondeterministic algorithms to proof systems is easy and we omit it here. Moreover, the corresponding results are due to Krajíček [6] who proved them by quite different means.

Definition 15. (1) Let $\mathbb{P}$ be a proof systems for Q. A sequence $(x_s)_{s\in\mathbb{N}}$ is *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 algorithm $\mathbb{W}$ with $\mathbb{P}(\mathbb{W}(1^s)) = x_s$ for all $s \in \mathbb{N}$.
(2) The problem Q *has hard sequences for proof systems* if every proof system for Q has a hard sequence.

For Q = TAUT the following result is already known (cf., e.g., the survey [2, Section 11]). We give a new proof that works for any, not necessarily paddable Π^p_t-complete problem Q.

Theorem 16. *Let Q be a Π^p_t-complete problem for some $t \geq 1$. Then:*

Q has no p-optimal proof system iff Q has hard sequences for proof systems.

Again an analysis of the proof of this theorem shows that for Q with padding, we can require that the claimed hard sequence $(x_s)_{s\in\mathbb{N}}$ satisfies $s \leq |x_s|$.

5 Hard Subsets

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 17. 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 $X \in \mathrm{P}$ and $\mathbb{A}$ is not polynomial time on X.
(2) The problem Q *has hard sets for (nondeterministic) algorithms* if every (nondeterministic) algorithm deciding Q has a hard set.

Using these notions the equivalences (a) $\Leftrightarrow$ (c) and (a) $\Leftrightarrow$ (d) in Theorem 12 (1) and (2), respectively, can be expressed in the following way:

Assume that Q has padding. Then
(1) Q has no almost optimal algorithm $\iff$ Q has hard sets for algorithms.
(2) Q has no almost optimal nondeterministic algorithm $\iff$ Q has hard sets for nondeterministic algorithms.

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

Corollary 18. *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 Π_t^p-complete, then Q has hard sequences for algorithms if and only if Q has hard sets for algorithms.

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 [10, 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. We write List(C, Q) 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 $\{L(\mathbb{M}_i) \mid i \geq 1\} = \{X \subseteq Q \mid X \in \mathrm{C}\}$.

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

Theorem 19. *(1) Q has hard sets for algorithms $\iff$ not* List(P, Q).

(2) Every nondeterministic algorithm $\mathbb{A}$ accepting Q is not polynomial on at least one subset X of Q with $X \in$ NP $\iff$ *not* List(NP, Q).

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

Definition 20. (1) Let $\mathbb{P}$ be a proof system for Q. A subset X of Q is *hard for $\mathbb{P}$* if $X \in$ P and there is no polynomial time algorithm $\mathbb{W}$ such that $\mathbb{P}(\mathbb{W}(x)) = x$ for all $x \in X$.

(2) *Q has hard sets for proof systems* if every proof system for Q has a hard set.

The following result can be obtained along the lines of the proof of Theorem 16.

Theorem 21. *Let Q be a problem with padding. Then:*

Q has no p-optimal proof system if and only if Q has 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.

Recall that 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.

Theorem 22. *(1) There exist problems in* $\mathrm{E} \setminus \mathrm{P}$ *with optimal algorithms.*
(2) If the Measure Hypothesis holds, then there exist problems in $\mathrm{NP} \setminus \mathrm{P}$ *with optimal algorithms.*

Here, $\mathrm{E} := \bigcup_{d\in\mathbb{N}} \mathrm{DTIME}(2^{d\cdot n})$. Messner [10] showed the existence of problems in $\mathrm{E} \setminus \mathrm{P}$ with *almost* optimal algorithms. 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 E = NE ($:= \bigcup_{d\in\mathbb{N}} \mathrm{NTIME}(2^{d\cdot n})$). We are able to show:

Theorem 23. *If the Measure Hypothesis holds, then there exist problems in* $\mathrm{NE} \setminus \mathrm{NP}$ *with optimal proof systems (or, equivalently, with almost optimal nondeterministic algorithms).*

Concerning algorithms which are not almost optimal but do not have hard sequences we derive the following results.

Theorem 24. *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 almost optimal and has hard sets but does not have hard sequences.*

The following example shows that the padding hypothesis is necessary in Theorem 24.

Example. Let $Q := \{1^n \mid n \in \mathbb{N}\}$. As $Q \in \mathrm{P}$, it has an almost optimal 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.

Corollary 25. *If the Measure Hypothesis holds, then the following are equivalent for $t \geq 1$:*
(i) No Π^p_t-complete problem has an almost optimal algorithm.
(ii) Every non-almost optimal algorithm deciding a Π^p_t-complete problem has hard sequences.

Theorem 26. *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.*

7 Getting Hard Sequences in an Effective Way

We have mentioned in the Introduction that Stockmeyer [13] has 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 his proof we derive a "nondeterministic" version.

Theorem 27. *Let Q be a* coNEXP*-hard problem. Then there is a polynomial time computable function $g : \Sigma^* \times \{1\}^* \to \Sigma^*$ such that for every nondeterministic algorithm $\mathbb{A}$ accepting Q the sequence $\big(g(\mathbb{A}, 1^s)\big)_{s\in\mathbb{N}}$ is hard for $\mathbb{A}$.*

Acknowledgments. The authors thank the John Templeton Foundation for its support through Grant #13152. 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). Moritz Müller thanks the FWF (Austrian Research Fund) for its support through Grant number P 23989 - N13.

References

1. Ben-David, S., Gringauze, A.: On the existence of optimal propositional proof systems and oracle-relativized propositional logic. Electronic Colloquium on Computational Complexity (ECCC), Technical Report TR98-021 (1998)
2. Beyersdorff, O.: On the correspondence between arithmetic theories and propositional proof systems - a survey. Mathematical Logic Quarterly 55(2), 116–137 (2009)
3. Chen, Y., Flum, J.: On p-Optimal Proof Systems and Logics for PTIME. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part II. LNCS, vol. 6199, pp. 321–332. Springer, Heidelberg (2010)
4. Chen, Y., Flum, J.: Listings and logics. In: Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS 2011), pp. 165–174. IEEE Computer Society (2011)
5. Hitchcock, J.M., Pavan, A.: Hardness hypotheses, derandomization, and circuit complexity. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 336–347. Springer, Heidelberg (2004)
6. Krajíček, J.: Bounded arithmetic, propositional logic, and complexity theory. Cambridge University Press (1995)
7. Krajíček, J., Pudlák, P.: Propositional proof systems, the consistency of first order theories and the complexity of computations. The Journal of Symbolic Logic 54, 1063–1088 (1989)
8. Köbler, J., Messner, J.: Complete problems for promise classes by optimal proof systems for test sets. In: Proceedings of the 13th IEEE Conference on Computational Complexity (CCC 1998), pp. 132–140 (1998)
9. Mayordomo, E.: Almost every set in exponential time is P-bi-immune. Theoretical Computer Science 136(2), 487–506 (1994)
10. Messner, J.: On the Simulation Order of Proof Systems. PhD Thesis, Univ. Erlangen (2000)
11. Monroe, H.: Speedup for natural problems and noncomputability. Theoretical Computer Science 412(4-5), 478–481 (2011)
12. Sadowski, Z.: On an optimal propositional proof system and the structure of easy subsets of TAUT. Theoretical Computer Science 288(1), 181–193 (2002)
13. Stockmeyer, L.: The Complexity of Decision Problems in Automata Theory. PhD. Thesis, MIT (1974)