Electronic Colloquium on Computational Complexity, Report No. 120 (2022)

On the existence of strong proof complexity generators

Jan Krajíček

Faculty of Mathematics and Physics
Charles University*

Abstract

The working conjecture from [8] that there is a proof complexity generator hard for all proof systems can be equivalently formulated (for p-time generators) without a reference to proof complexity notions as follows:

- There exist a p-time function g extending each input by one bit such that its range $rng(g)$ intersects all infinite NP sets.

We consider several facets of this conjecture, including its links to bounded arithmetic (witnessing and independence results) and the range avoidance problem, to time-bounded Kolmogorov complexity, to feasible disjunction property of propositional proof systems and to complexity of proof search. We argue that a specific gadget generator from [10] is a good candidate for g.

Keywords: proof complexity generators, weak pigeonhole principle, range avoidance problem, time-bounded Kolmogorov complexity, proof search, feasible disjunction property.

1 Introduction

As pointed out by Cook and Reckhow [3], the *NP vs. coNP problem* (asking whether the computational complexity class NP is closed under complementation) can be equivalently restated as a question whether all propositional tautologies have fixed-polynomial size proofs in some propositional proof system. A **propositional proof system** (to be abbreviated *pps*) in their sense is a p-time decidable binary relation $P(x, y)$ such that $\exists x P(x, y)$ defines exactly TAUT, the set of tautologies in the DeMorgan language.

A pps P does not admit such short proofs (i.e. witnesses x) iff there exists a subset $H \subseteq$ TAUT such that for any $c \geq 1$, only finitely many $\tau \in H$ have P-proofs of size bounded above by $|\tau|^c$; that is, $\exists x(|x| \leq |y|^c)P(x, y)$ is satisfied

*Sokolovská 83, Prague, 186 75, The Czech Republic, `krajicek@karlin.mff.cuni.cz`

ISSN 1433-8092

by a finite number of elements of H only. Any such set H will be said to be **hard for** P.

There are essentially only two classes of formulas known that make plausible candidates for being hard for strong pps: reflection principles and τ-formulas coming from proof complexity generators. The former class is a classic topic of proof complexity and its exposition can be found in [14, Sec.19.2].

The latter formulas are constructed as follows. Take a function $g : \{0,1\}^* \rightarrow \{0,1\}^*$ that stretches all size n inputs to size $m = m(n) > n$ (and hence the complement of its range $rng(g)$ is infinite) and such that its restriction g_n to $\{0,1\}^n$ is computed by a size $m^{O(1)}$ circuit C_n. For each $b \in \{0,1\}^m \setminus rng(g_n)$ encode naturally (as in the NP-completeness of SAT) the statement

$$|x| = n \rightarrow C_n(x) \neq b$$

by a size $m^{O(1)}$ tautology $\tau(g)_b$. Function g is said to **hard for** P iff the set $\bigcup_{n\geq 1}\{\tau(g)_b \mid b \in \{0,1\}^{m(n)} \setminus rng(g_n)\}$ is hard for P, and we speak of function g as of a **proof complexity generator** in this context.

We shall actually restrict ourselves here[1] on the rudimentary case of generators g computed in *uniform p-time* (except the example of function $\mathbf{tt}_{s,k}$ that is computed in uniform time $m^{O(1)}$) and, in fact, Lemma 6.3 shows that non-uniformity is to some extent irrelevant.

The $\tau(g)_b$-formulas were defined in [6] and independently and with an apparently different motivation in Alekhnovich et.al. [1]. Unfortunately the authors of [1] did not pursue the topic[2]. The theory of proof complexity generators thus grew out of the motivation for the formulas in [6]: a logic question about the provability of the dual weak PHP (dWPHP) for p-time functions in weak bounded arithmetic theories PV and S^1_2, cf. [6, Problem 7.7]. The dWPHP(f) says that function f does not map any interval $[0,a]$ onto $[0,2a]$ (the term $2a$ can be altered to various other values, e.g. to a^2 etc., without changing the logical strength of the principle).

The theory has now several facets connecting it to various parts of logic and of proof and computational complexity. The interested reader may look at [14, Sec.19.4-6] or at older [11, Chpts.29-30] for an overview and further references.

Right from the beginning there were two working conjectures:

1. *There are generators pseudo-surjective for Extended Frege systems EF*, cf.[6, Conj.7.9, Cor.7.10],[7, Conj.4.1,Cor.4.2].

   This conjecture is related to the provability problem mentioned above (and also to the hardness of function $\mathbf{tt}_{s,k}$ defined below) and the pseudo-surjectivity (and related notions of iterability and freeness) implies the hardness. We shall touch upon it in Section 2, details are in [7, 8].

[1]Note that one can allow that output bits of generator g are computed in non-uniform $NTime(m^{O(1)}) \cap coNTime(m^{O(1)})$ and still get tautologies of size polynomial in m expressing that $b \notin rng(g_n)$, cf. [17, Conj.2], [13, Conj.1] and [9, 12]. There are quite a few facts known about such generators and the interested reader may start with [9, 12, 13].

[2]With the sole exception of Razborov [17] written in 2002/03.

2. *There is a generator hard for all proof systems*, cf.[8, Sec.2].

We shall concentrate here on the second conjecture and we shall restrict our attention to uniform generators having the minimal required stretch $m(n) = n + 1$. It is easy to see that truncating any p-time generator to output-size $n+1$ preserves the hardness (over any pps simulating resolution). It also allows for a particularly simple formulation of Conjecture 1.1: by [8, Sec.1] (or [14, L.19.4.1]) the second conjecture can be then restated without any reference to proof complexity notions as follows.

Conjecture 1.1 ([8, Sec.2])
There exist a p-time function g extending each input by one bit such that its range $rng(g)$ intersects all infinite NP sets. That is, the complement of $rng(g)$ is NP-immune.

Let us remark that the candidate hard generator proposed in [8] had stretch $n + 1$ and was in non-uniform p-time: in that case one needs to allow in the formulation above also NP/poly sets.

With a bit of imagination one can interpret the conjecture as an extension of the paradigm of the feasible interpolation method from proof complexity (cf. [14, Chpts.17 and 18]) to all proof systems:

- *short proofs* (here witnesses to the membership in an infinite NP set A, i.e. p-size proofs in some proof system)
- *imply an upper bound on some computational resource* (here a non-trivial upper bound on compression, using g as decompressing algorithm).

An illuminating example of a possibly strong generator is the **truth-table function** $\mathbf{tt}_{s,k}$ sending a size s circuit in k inputs to its truth-table (a size 2^k string), cf.[8] or [14, 19.5]. Circuits of size s can be coded by $10s \log s$ bits and so to make the function stretching we assume that $n := 10s \log s < m(n) := 2^k$ (hence size s circuits are coded by $n < m$ bits). It is computed in (uniform) time $O(sm) = 2^{O(k)}$, so it is p-time if $s = 2^{\Omega(k)}$.

The τ-formulas determined by this generator state circuit lower bounds for particular Boolean functions: $\tau(\mathbf{tt}_{s,k})_b \in \text{TAUT}$ iff the function with truth-table b has circuit complexity bigger than s. This makes the formulas attractive but also hard to approach as we know very little about size of general circuits.

It is known that if a pps P admits any pseudo-surjective function (cf. Section 2) then $\mathbf{tt}_{s,k}$ is pseudo-surjective too (the rate of s depends on the rate of lower bound for P establishing the pseudo-surjectivity) and hence hard for P as well, cf.[8] or [11, Sec.30.1]. The first working conjecture above thus implies that the τ-formulas determined by the truth table function are hard for EF. On the other hand, unless $\text{NE} \cap \text{coNE} \subseteq \text{P}/poly$, this generator cannot be hard for all proof systems[3] and hence it is not a good candidate for Conjecture 1.1, cf. [11, p.198].

[3]But to find a pps for which it is not hard is likely to be a hard task itself, cf.[11, L.29.2.2].

Our second example follows [16, Remark 6.1] and concerns time-bounded Kolmogorov complexity. Recall that the complexity measure $K^t(w)$ is the minimal size of a program that prints w in time at most $t(|w|)$, cf. Allender [2]. The point is that a proof complexity generator with a larger stretch $m >> n$ produces strings of smaller than maximal K^t complexity. For example, if g stretches n bits to $2n$ bits and runs in time $t(n) = n^{O(1)}$ then for all size $2n$ strings $w \in Rng(g_n)$ and $n >> 0$:

$$K^t(w) \leq n + O(1) < 2m/3 \ .$$

In fact, as discussed in [16, 6.1], for a fixed polynomial time $t(n)$ sufficient for the computation of g one can consider the universal Turing machine U^t underlying the definition of K^t as a generator itself[4]. Then for any pps P simulating EF, if some $\tau(U^t)$-formulas have short P-proofs (e.g. by proving tautologies expressing the lower bound $K^t(w) \geq 2m/3$) so do some $\tau(g)$-formulas. That is, if there is any g computable in time t and hard for P then U^t must be hard as well.

In this paper we consider first a couple of ways how one could try to disprove (or at least limit possible g in) Conjecture 1.1: finding a feasible way to witness that the complement of $rng(g)$ is non-empty (Section 2) or reducing the question of possible stretch to the task to prove lower bounds on time-bounded Kolmogorov complexity (Section 3). We argue that known results imply that neither of these approaches is likely to work without proving first super-polynomial lower bounds for (uniform and non-uniform) computations. We also indicate in Section 4 how to modify the notion of a generator (and the conjecture and results in Sections 2 and 3) to address the hardness of proof search instead of lengths-of-proofs.

In Section 5 we discuss a new definition of hardness, the $\bigvee$-hardness, that strengthens (presumably) the hardness as defined above but is weaker (also presumably) than notions of pseudo-surjectivity and iterability. The reason for introducing the new notion is that a particular generator from the class of gadget generators introduced in [10] is the $\bigvee$-hardest[5] among all generators but (presumably) not under the hardness as defined above: in [10] we used for this result the iterability as it was at hand but that is not good for Conjecture 1.1. Namely, it is known (cf. [8]) that if there is any iterable (or even pseudo-surjective) map then $\mathbf{tt}_{s,k}$ is iterable (or pseudo-surjective) too and hence hard. But by the remark above $\mathbf{tt}_{s,k}$ is unlikely to be hard for all proof systems.

This new notion of $\bigvee$-hardness is equivalent to the hardness as defined above for a class of pps satisfying the strong feasible disjunction property (Section 5). This class has the property that all pps *not in it* are automatically not p-bounded. In Section 6 we put forward a specific uniform gadget generator computed in sub-quadratic time. In Section 7 we discuss two ways how to scale the conjecture and we show, under a hypothesis, that Conjecture 1.1 holds

[4]A similar observation was made recently in Ren, Santhanam and Wang [18].

[5]Ren, Santhanam and Wang [18] speak informally about the hardest proof complexity generator but what they define is formally an infinite family of generators.

relative to all *feasibly infinite* NP sets: sets for which there is a p-time function picking arbitrarily large elements of the set. The paper is concluded by a remark in Section 8.

Basic proof complexity background can be found in [14, Chpt.1], the topic of hard formulas (including a brief introduction to the theory of proof complexity generators) is in [14, Chpt.19]. When we use some proof complexity notions and facts in formal statement we define them first (and give a reference). But we also use proof complexity background in various informal remarks and there we only refer to the original source and/or to a place in [14] where it can be found.

2 Witnessing the dWPHP aka the range avoidance problem

The dWPHP for function g extending n bits to $m = m(n)$ bits is formalized by the formula

$$\forall 1^{(n)} \exists y(|y| = m) \forall x(|x| = n)\ g(x) \neq y\ .$$

To witness this formula means to find a witness y for the existential quantifier given as input $1^{(n)}$. This task became known recently as the *range avoidance problem*[6].

Witnessing is a classic notion of proof theory[7] and, in particular, many fundamental results in bounded arithmetic are formulated as follows: if theory T proves a formula of syntactic complexity S then it can be witnessed (i.e. its leading $\exists$ can be witnessed) by a function from computational class C. Such statements are known[8] for many basic bounded arithmetic theories, many natural syntactic classes of formulas and computational classes of functions.

Unprovability results are generally very difficult and usually conditional, and we shall use one below. But in the relativised set-up (in our situation this would mean that g is given by an oracle) many unconditional unprovability results are known and they are usually derived by showing that a principle at hand cannot be witnessed by a function in some class C (for dWPHP see the end of this section).

We now give an application of the conditional unprovability result of [15]. Consider theory T_{PV} whose language has a k-ary function symbol f_M attached to every p-time clocked machine M with k inputs, all $k \geq 1$. Symbol f_M is naturally interpreted on $\mathbf{N}$ by the function M computes. The axioms of T_{PV} are all universal sentence in the language true in $\mathbf{N}$ under this interpretation.

[6]That problem deals with functions computed by non-uniform circuits but that is included in the formulation above as g can have parameters (not shown in the notation).

[7]In particular, witnessing of dWPHP is discussed in [8, Sec.7].

[8]There are many precise statements about the (mutual) provability of combinatorial principles of various complexities in bounded arithmetic theories in terms of witnessing, reducibilities among them (corresponding to provability over various weak theories) and complete problems in such classes. For reasons that I do not quite understand complexity theorists prefer to ignore this knowledge and rediscover some of it again in a language avoiding but still simulating logic concepts.

The hypothesis used in the unprovability result is this.

Hypothesis (H):

There exists constant $d \geq 1$ such that every language in P can be decided by circuits of size $O(n^d)$: $P \subseteq Size(n^d)$.

The possibility that (H) is true with $d = 1$ is attributed to Kolmogorov but it is not a hypothesis accepted by mainstream complexity theory. However, there are no technical results supporting the skepticism and, in fact, (H) has a number of great consequences (as is P $\neq$ NP, see [15] for further, including proof complexity).

We shall use $g := \mathbf{tt}_{s,k}$ with $s = 2^{\epsilon k}$ for a fixed $0 < \epsilon < 1$ for our p-time function[9].

Theorem 2.1 ([15])

Assume hypothesis (H). Then for every $0 < \epsilon < 1$ and $s = s(k) := 2^{\epsilon k}$ the theory T_{PV} does not prove the sentence

$$\forall 1^m (m = 2^k > 1) \exists y \in \{0,1\}^m \forall x \in \{0,1\}^n,\ \mathbf{tt}_{s,k}(x) \neq y \tag{1}$$

expressing the dWPHP for $\mathbf{tt}_{s,k}$, where $n := 10 s \log s$.

The reason why this rules out witnessing dWPHP in p-time is that the property that f witnesses dWPHP is itself a universal statement

$$\forall 1^{(n)} \forall x (|x| = n)\ |f(1^{(n)}| = m \wedge g(x) \neq f(1^{(n)})$$

and hence already an axiom of T_{PV}. In fact, we can consider an interactive model of witnessing via constant round Student - Teacher computation and it still cannot witness dWPHP. In this computation p-time student S, given $1^{(n)}$, produces his candidate solution $b_1 \in \{0,1\}^m$. Computationally unlimited teacher T either acknowledges the correctness or she produces a counter-example: $x_1 \in \{0,1\}^n$ s.t. $g(x_1) = b_1$. S then produces his second candidate solution b_2 using also x_1, T either accepts it or gives counter-example x_2 etc. The requirement is that within a given bound t on the number of rounds S always succeeds. This can be written in a universal way as

$$g(x_1) \neq S(1^{(n)}) \vee g(x_2) \neq S(1^{(n)}, x_1) \vee \ldots \vee g(x_n) \neq S(1^{(n)}, x_1, \ldots, x_{t-1})\ . \tag{2}$$

Let us remark that S-T protocol with polynomially many rounds $t = m^{O(1)}$ corresponds to the notion of pseudo-surjectivity mentioned in the Introduction[10] while $O(1)$ rounds correspond to the notion of freeness: the universal statement (2) can be represented by an infinite family of p-size tautologies and the two hardness notions require that these tautologies do not have short proofs, cf.[7, 8] for details.

Let us state the conclusion of this discussion formally.

[9] As pointed out in [15], the theorem holds for the gadget generator (cf. Section 6) too.

[10] Iterability further restricts possible S.

Theorem 2.2

Assume hypothesis (H). Then dWPHP for function $\mathbf{tt}_{s,k}$ *with parameters as in Theorem 2.1 can be witnessed neither by a p-time function nor by a Student-Teacher computation with p-time Student and constantly many rounds.*

Hence to disprove the conjecture by witnessing feasibly the non-emptiness of the complement of $rng(g)$ for all p-time g would imply super-polynomial lower bounds for circuits.

If we manage to extend the unprovability to theory $T_{\mathrm{PV}} \cup S^1_2$ then we would rule out witnessing by S-T computation with polynomially many rounds. Extending it further to theory $T_{\mathrm{PV}} \cup T^1_2$ (or equivalently to $T_{\mathrm{PV}} \cup S^2_2$) would rule out witnessing by p-time machines accessing an NP oracle. All these result need to be conditional as they imply (unconditionally) that P differs from NP: if P = NP then this is implied by a true universal statement in the language of T_{PV} (saying that a particular p-time algorithm solves SAT) and hence all true universal closures of bounded formulas are equivalent over T_{PV} to universal statements and hence in T_{PV}.

Further note that in the relativised world we have a number of unconditional results about the impossibility to witness dWPHP. As an example let us mention that we cannot witness by non-uniform p-time machine with an access to an NP^R oracle where R is the graph of g that g is not a bijection between $[0, a]$ and $[0, 2a]$. Another example is that even if we have oracle access to g and to another function f we cannot witness by a PLS problem with base data defined by p-time machines with oracle access to f, g that g is not a bijection between $[0, a]$ and $[0, 2a]$ with f being its inverse map. The interested reader can find these results (and all background) in [5, Secs.11.2-3] and in references given there.

3 Stretch and the Kt-complexity

The main aim of proof complexity generators is to provide hard examples and for this purpose the stretch $n + 1$ of g in Conjecture 1.1 suffices (and it yields the shortest τ-formulas). A larger stretch is of interest in a connection[11] with the truth-table function $\mathbf{tt}_{s,k}$ discusses earlier.

We may try to limit possible stretch of generators via some considerations involving time-bounded Kolmogorov complexity as we touched upon in the Introduction. We shall use Levin's measure $Kt(w)$: the minimum value of $|d| + \log t$, where program d prints w in time t, cf. Allender [2]. Its advantage over K^t is that it does not require to fix the time in advance. Although statement like $Kt(w) \geq 2m/3$ cannot be presumably expressed by a p-size (in m) tautology if we had an NP set A all of which members $w \in A$ satisfy $Kt(w) \geq 2|w|/3$ then certificates for the membership of w in A can be interpreted as proofs of $Kt(w) \geq 2m/3$.

[11] In fact, the need for larger stretch even in this connection seems to be eliminated by the notion of iterability, cf. [8].

Let us consider a function with an extreme stretch: $\mathbf{tt}_{s,k}$ with $s = 100k$. This generator sends $n = 10s \log s \leq O(\log m \log\log m)$ bits to $m = 2^k$ bits and is computed in time $t = O(sm) < m^{3/2}$. Hence both K^t (here $t(n) = n^{3/2}$ suffices) and Kt are bounded above on $rng(\mathbf{tt}_{s,k}) \cap \{0,1\}^m$ by $O(\log m \log\log m)$.

Notation (Allender [2]):
For any set $A \subseteq \{0,1\}^*$ define function $Kt_A : \mathbf{N}^+ \to \mathbf{N}^+$ by

$$Kt_A(m) \ := \ \min\{Kt(w) \mid w \in \{0,1\}^m \cap A\}$$

if the right-hand side is non-empty, and we leave $Kt_A(m)$ undefined otherwise.

Hence we could limit the extreme stretch as above if we could find an infinite NP set A such that $Kt_A(m) \geq \omega(\log m)$. Unfortunately the next theorem suggests that this is likely not an easy task. Following Allender [2] we define an **NE search problem** to be a binary relation $R(x, y)$ such that R implicitly bounds $|y|$ by $2^{O(n)}$ for $|x| = n$ and which is decidable in time $2^{O(n)}$ (think of y as an accepting computation of an NE machine on input x). The search task is: given x, find y such that $R(x, y)$, if it exists. As an example related to our situation consider $R(x, y)$ defined by:

$$\exists y(|y| = x \wedge y \in A$$

where $A \in \mathrm{NP}$.

Theorem 3.1 (Allender [2, Cor.7,Thm.8])
There exists an infinite NP set A s.t. $Kt_A(m) = \omega(\log m)$ iff there exists an NE search problem s.t.:

- *$\exists y R(x, y)$ is satisfied for infinitely many x,*
- *every algorithm running in time $2^{O(n)}$ solves the search problem for a finite number of inputs x only.*

Hence ruling out generators with even very large stretch means likely to prove significant computational lower bounds. The following seems to be a natural test question whether anything can be proved via this approach at all.

Problem 3.2
Is it true that any infinite NP set A contains a string $w \in A$ with $Kt(w) < |w|$? That is, is it true that the set $\{w \mid Kt(w) \geq |w|\}$ is NP-immune?

The negative answer to the problem would rule out generators g in Conjecture 1.1 stretching n bits to $n + \omega(\log n)$ bits. We would rather like to see the affirmative answer as it is in the spirit of the remark after Conjecture 1.1.

4 A modification for proof search hardness

Proof complexity generators, and Conjecture 1.1 in particular, aim primarily at the problem to establish lengths-of-proofs lower bounds. It is easy to modify the concept to aim at time complexity of proof search. Essentially this means to replace everywhere in the previous sections NP sets by P sets. To give a little more detail we shall use the definition of a proof search algorithm from [16]: it is a pair (A, P) such that A is a deterministic algorithm that finds for every tautology its P-proof. How much time any algorithm (A, P) has to use on a particular tautology is measured by the information efficiency function $i_P : \mathrm{TAUT} \rightarrow \mathbf{N}^+$; it is an inherently algorithmic information concepts. For each pps P there is a time-optimal (A_P, P) which is also information-optimal. The reader can find definitions and proofs of these facts in [16].

Define a set $S \subseteq \mathrm{TAUT}$ to be **search-hard for** P iff for any $c \geq 1$ algorithm A_P finds a proof of σ in time bounded above by $|\sigma|^c$ for finitely many formulas $\sigma \in S$ only. Then analogously with the definition of hardness we define g (in the format as in Conjecture 1.1, i.e. p-time stretching each input by one bit) to be **search-hard for** P iff the complement of $rng(g)$ is search-hard for P. It is easy to see that the conjecture that there is a uniform generator search-hard for all pps is then equivalent to

Conjecture 4.1 (proof search version of Conjecture 1.1)
There exist a p-time function g extending each input by one bit such that its range $rng(g)$ intersects all infinite P sets. That is, the complement of $rng(g)$ is P-immune.

There are some more facts known about Kt_A measure for sets in P. We mention two.

Theorem 4.2

1. **(Allender [2, Thms.6,8])**

   For time $t(n)$, there exists $A \in P$ s.t. $Kt_A(m) \geq \omega(\log t(m^{O(1)})$ iff there exists an NE search problem $R(x, y)$ satisfied for infinitely many x s.t. any algorithm running in time $t(2^{O(n)})$ solves it for a finitely many inputs x only.

   In particular, $Kt_A(m) = \omega(\log m)$ for a set $A \in P$ iff there exists an NE search problem that is satisfied for infinitely many x but every algorithm running in time $2^{O(n)}$ solves the search problem for a finite number of inputs x only.

2. **(Hirara [4, Thm.3.11])**

   There exists a (time-constructible) function $s(n) \leq n - \omega(\log n)$ s.t. the set

$$\{w \mid Kt(w) \geq s(|w|)\}$$

is P-immune iff there exists a SAT algorithm for each P-uniform sequence of circuits that runs in time $2^n/n^{\omega(1)}$.

(This is a special case of [4, Thm.3.11] only.)

5 Feasible disjunction property and $\bigvee$-hardness

We shall propose in this section a notion of hardness that is preserved by more constructions (and, in particularly, by the construction underlying gadget generators in Section 6) than is the original hardness but is presumably weaker than the notion of iterability used in [10], as it was discussed in the Introduction.

Definition 5.1
A function $g : \{0,1\}^* \rightarrow \{0,1\}^*$ that stretches all size n inputs to size $m := m(n) > n$ and is computed by size $m^{O(1)}$ circuits is **$\bigvee$-hard** *for a pps P is for any $c \geq 1$, only finitely many disjunctions*

$$\tau(g_n)_{b_1} \vee \ldots \vee \tau(g_n)_{b_r} \ ,$$

$n \geq 1$ and all $b_i \in \{0,1\}^m$, have P-proof of size at most m^c.

A pps P has the **feasible disjunction property** (abbreviated *fdp*) iff whenever a disjunction $\alpha_0 \vee \alpha_1$ of two formulas having no atoms in common has a P-proof of size s then one of α_i has a P-proof of size $s^{O(1)}$. The **strong fdp** is defined in the same way but the starting disjunction can have any arity r: $\bigvee_{i<r} \alpha_i$. The strong fdp property plays a role in analysis of a proof complexity generator in [12], see also [14, Subsec.17.9.2]. It is an open problem ([14, Prob.17.9.1]) whether, for example, Frege or Extended Frege systems have the (strong) fdp. It does not seem likely but a pps P proving shortly its own reflection principles (as Frege and Extended Frege systems do) and having the fdp has the remarkable property that for any tautology α, P either proves shortly α or it proves shortly that *it does not prove shortly* α.

Lemma 5.2 *Assume a pps P has the strong fdp. Then any generator hard for P is also $\bigvee$-hard for P.*

A strategic choice: use $\bigvee$-hardness
As it was pointed out in [12], for the purpose of proving lengths-of-proofs lower bounds for some pps P we may assume w.l.o.g. that P satisfies the strong fdp property: otherwise it is not p-bounded and we are done. This observation, together with Lemma 5.2, justifies the use of $\bigvee$-hardness rather than mere hardness.

The reader skeptical to the choice may interpret the statements contrapositively as sufficient conditions refuting the strong fdp for a particular pps, cf. Lemma 6.4. In particular, it may happen that no strong pps has the strong fdp: but then we can celebrate as NP $\neq$ coNP.

Let us conclude this section by noticing that the fdp property can be naturally modified for proof search as well: time A_P needs on α_0 or α_1 is bounded above by a polynomial in time it needs on $\alpha_0 \vee \alpha_1$. However, such a property implies the usual feasible interpolation property. Namely, if π is a P-proof of a disjunction

$$\gamma_0(x, y) \vee \gamma_1(x, z)$$

(the disjuncts are not required to have disjoint sets of variables this time) consider disjunction $\beta \vee (\gamma_0 \vee \gamma_1)$ where β is the conjunctions of 0 with all bits of π. Then A_P when given this disjunction reads π and hence proves $\gamma_0 \vee \gamma_1$ and thus also $\beta \vee (\gamma_0 \vee \gamma_1)$. By the search-version of fdp A_P must find in time polynomial in $|\pi|$ a proof of $\gamma_0 \vee \gamma_1$ (as β is false) and thus also of any instance $\gamma_0(a, y) \vee \gamma_1(a, z)$ (this requires that P-proofs are in some sense closed under substitution of constants). By the new property again algorithm A_P, for each a succeeds on either $\gamma_0(a, y)$ or on $\gamma_1(a, z)$ in time polynomial in $|\pi|$. That yields feasible interpolation. This observation means that the proof search variant of fdp cannot hold for any strong proof systems and is subject to same limitations as is feasible interpolation and, in particular, cannot hold for any strong proof systems unless some standard cryptographic assumptions fail. The reader can find all background in [14].

6 The gadget generator

The class of gadget generators was introduced in [10] and it is defined as follows. Given any p-time function

$$f \ : \ \{0,1\}^\ell \times \{0,1\}^k \ \rightarrow \ \{0,1\}^{k+1}$$

define a **gadget generator based on f**

$$Gad_f \ : \ \{0,1\}^n \rightarrow \{0,1\}^m$$

where

$$n := \ell + k(\ell + 1) \ \text{ and } \ m := n + 1$$

as follows:

1. *The input $\overline{x} \in \{0,1\}^n$ is interpreted as $\ell + 2$ strings*
$$v, u^1, \dots, u^{\ell+1}$$
where $v \in \{0,1\}^\ell$ and $u^i \in \{0,1\}^k$ for all i.

2. *The output $\overline{y} = Gad_f(\overline{x})$ is the concatenation of $\ell + 1$ strings $w^s \in \{0,1\}^{k+1}$ where we put*
$$w^s \ := \ f(v, u^s) \ .$$

Clearly we may fix f w.l.o.g. to be the **circuit value function** $CV_{\ell,k}(v,u)$ which from a size ℓ description v of a circuit (denoted also v) with k inputs and $k+1$ outputs and from $u \in \{0,1\}^k$ computes the value of v on u, an element of $\{0,1\}^{k+1}$.

It was shown in [10] (see also [14, L.19.4.6]) that if we replace the hardness of a generator by a stronger condition then it suffices to consider circuits v of size $\leq k^{1+\epsilon}$, any fixed $\epsilon > 0$. The proof of this fact in [10] used the notion of iterability mentioned earlier, as it was at hand. However, the same argument gives Theorem 6.1 using a presumably weaker notion of $\bigvee$-hardness from Section 5; we shall not repeat the argument.

Theorem 6.1
Assume that there exists a p-time function $g : \{0,1\}^* \rightarrow \{0,1\}^*$ that stretches all size n inputs to size $m := m(n) > n$ that is $\bigvee$-hard for a pps P.

Then the gadget generator based on $CV_{k^2,k}$ is $\bigvee$-hard (and hence also hard) for P as well.

Notation:
In the rest of the note we shall ease on the notation and we will denote the gadget generator Gad_f based on $f = CV_{k^2,k}$ by Gad_{sq}.

Note that a circuit of size s can be encoded by $10s \log s$ bits so Gad_{sq} uses as gadgets circuits of size little bit less than quadratic. Observe also the following simple statement.

Lemma 6.2 *Gad_{sq} is computed in time smaller than $n^{3/2}$.*

The next statement shows that non-uniformity is irrelevant in the presence of strong fdp. It is proved by taking for gadgets circuits of the size needed to compute the generator.

Lemma 6.3 *Assume a pps P admits a $\bigvee$-hard proof complexity generator computed in non-uniform p-time (i.e. by p-size circuits). Then Gad_{sq} is $\bigvee$-hard for P.*

Let us consider the stretch of gadget generators. By default it was taken in the definition to be the minimal required stretch but there are other options. One could use as gadgets circuits that map k bits to k' bits where $k' >> k$; for example, $k' = 2k$ or $k' = k^2$ (allowing accordingly a bigger size of gadgets, still polynomial in k). The resulting generator would send n bits to approximately $(k'/k)n$ bits which is about $n^{1+\epsilon}$ for some $\epsilon > 0$.

However, we want to be conservative with requirements on gadgets. Note that the stretch of gadget generators can be influenced also by taking more strings u^i in the construction of Gad_f than is the minimal number needed, i.e. more than $\ell + 1$. In particular, assume we perform the construction of Gad_f but taking $t > \ell$ string u^i and w^i. We still want to maintain, as in Theorem 6.1, that the generator is the hardest $\bigvee$-hard generator; hence we still allow only t polynomial in k.

Then

$$n := \ell + kt \ \text{ and } \ m := (k+1)t \ .$$

For $\ell \leq k^{O(1)}$ (as in Gad_{sq}) and taking $t := k^c$ for very large $c \geq 1$ we can arrange that

$$m \ \geq \ n + n^{1-\epsilon}$$

for as small $\epsilon > 0$ as wanted. Denote the generator which extends the definition of Gad_{sq} in this way simply Gad^c_{sq}.

Lemma 6.4

Assume that there is an infinite NP set A such that for some $\delta > 0$:

$$K^t_A(m) \geq m - m^{1-\delta} \ ,$$

where $t(n) = n^{3/2}$. Assume further that Conjecture 1.1 is true.

Then there is a pps P such that no pps Q simulating P has the strong fdp.

Proof :

Choose $c \geq 1$ so large that the stretch of Gad^c_{sq} is $n^{1-\epsilon}$ where

$$n^{1-\epsilon} >> m^{1-\delta} = (n + n^{1-\epsilon})^{1-\delta}$$

for $n >> 0$ (taking $c \geq 1$ such that $\delta < \epsilon < 1$ suffices).

Given an infinite NP set A satisfying the hypothesis define a pps P to be, say, resolution but accepting also witnesses to the membership of $b \in A$ as proofs of $\tau(\mathrm{Gad}^c_{sq})_b$.

If Conjecture 1.1 was true for any Q simulating this P and Q would satisfy the strong fdp, it would follow by Theorem 6.1 (modified trivially for Gad^c_{sq}) that Gad^c_{sq} is hard for Q. That is a contradiction with how P was defined.

q.e.d.

It is known that gadget generators (and Gad_{sq} in particular) are hard for many proof systems for which we know any super-polynomial lower bound, cf.[14]. Informally, the advantage of working with Gad_{sq} is that instead of proving hardness for specific gadget it suffices to show that it cannot be shortly disproved that a hard gadget exists.

Our **working hypothesis** is that generator Gad_{sq} satisfies Conjecture 1.1. But when working with the generator we encounter the same difficulty as in the case of the truth-table generator $\mathbf{tt}_{s,k}$: we know nothing non-trivial about circuits of sub-quadratic size. Furthermore, the experience with lengths-of-proofs lower bounds we have so far suggests that it is instrumental to have hard examples with some clear combinatorial structure. Hence to study the hardness of Gad_{sq} it may be advantageous to consider gadgets (i.e. sub-quadratic circuits) of a special form (technically that would be a substitution instance of Gad_{sq}).

One such specific generator was defined in [14, pp.431-2] and denoted $\mathrm{nw}_{k,c}$ there; its gadget is essentially a slightly over-determined system of sparse equations for a generic function. Namely, gadget consists of $k+1$ sets $J_1, \dots, J_{k+1} \subseteq \{x_1, \dots, x_k\}$, each of size $1 \leq c \leq \log k$, together with 2^c bits defining truth table of a Boolean function f with c inputs. Given gadget u and $v \in \{0,1\}^k$, $f(u,v) \in \{0,1\}^{k+1}$ are the $k+1$ values f computes on values that v gives to variables in sets $J_1, \dots, J_{k+1}$. This generator for one fixed, non-uniform gadget was the original suggestion for Conjecture 1.1 in [8] but the gadget generator construction allows to leave the non-uniformity and consider generic case.

7 Scaling the conjecture

As mentioned in the introduction, Conjecture 1.1 for g is equivalent to the statement that g is hard for all pps P. A natural scaling of the conjecture is thus by considering it for specific pps P. In the language of Conjecture 1.1 this amounts to restricting to NP sets A from the class of those for which P can shortly prove (the tautologies expressing for all lengths $n \geq 1$) that $A \cap rng(g) = \emptyset$. This class is the resultant Res_g^P as defined in [8] and the reader can find details there.

However, there is another scaling possible. Given a sound theory T (whose language extends that of T_{PV} for convenience) consider the class of all NP sets A such that the infinitude of A:

$$Inf_A \ := \ \forall x \exists y (y > x \wedge y \in A)$$

can be proved in T. Formula $y \in A$ can be written as a bounded existential formula in the language of T_{PV} and Inf_A is thus an $\forall\exists$-sentence.

Knowing that a particular T proves Inf_A yields, in principle, a non-trivial information about A. For example, if T_{PV} proves the sentence then by Herbrand's theorem there is a p-time function f witnessing it. That is, f finds elements of A:

$$\forall x (f(x) > x \wedge f(x) \in A) \ .$$

We shall call sets A for which such p-time function f exists **feasibly infinite**. This remains true (by Buss's theorem) if T_{PV} is augmented by S^1_2. If T_{PV} is extended by some stronger bounded arithmetic theory then Inf_A will be witness be a specific NP search problem attached to the theory. For example, if we add to T_{PV} induction axioms for NP sets (theory T^1_2) then Inf_A is witnessed by a PLS problem (by the Buss-K. theorem). The reader can find the bounded arithmetic background in [5].

It is easy to see that Problem 3.2 has the affirmative answer for feasibly infinite NP sets. For the conjecture we need to work a bit.

Theorem 7.1

Assume hypothesis (H) from Section 2. Then Conjecture 1.1 holds relative to the class of feasibly infinite NP sets: there is a generator g whose range intersects every feasibly infinite NP set.

Proof :

The proof is a special case of the construction from [15]. We shall show that generator $\mathbf{tt}_{s,k}$ with $s = s(k) := 2^{k/2}$ satisfies the statement.

Let A be a feasibly infinite NP set as it is witnessed by a p-time function f. Let $d \geq 1$ be the constant from (H) and put $m' := m^{1/(3d)}$ where $m := |f(1^n)|$ and $n >> 1$, and put also $k := \log m$.

Define the function $\hat{f}$ that has $m' + k$ variables and on inputs $1^{m'}$ and $i \in \{0,1\}^k$ computes the i-th bit of $f(1^n)$; it is a p-time function.

Take a circuit $\hat{C}(z,i)$ that computes $\hat{f}$ of size guaranteed by hypothesis (H) and define new circuit C by substituting $1^{m'}$ for z in $\hat{C}$ and leaving only the k variables for bits of i. Note that C has size $O((m'+k)^d) < 2^{k/2}$. Further, by its definition, $\mathbf{tt}_{s,k}(C) = f(1^n)$; i.e. $rng(\mathbf{tt}_{s,k}) \cap A \neq \emptyset$.

q.e.d.

8 Concluding remarks

I think that it is fundamental for further development of the theory to make a progress on the original problem[12] of the unprovability of dWPHP for p-time functions discussed in the Introduction. For a start we may try to show the unprovability in $T_{\rm PV}$ (or some of its extension as mentioned at the end of Section 2) under a more mainstream hypothesis than is (H).

However, real progress will result only from unconditional results. For reasons discussed at the end of Section 2 to have a chance to succeed we need to leave theory $T_{\rm PV}$ aside and work with theories PV or S^1_2. This implies that an argument cannot rely just on witnessing theorems as they do not change if $T_{\rm PV}$ is added. The problem becomes essentially propositional and it is exactly this what led to notions of freeness and pseudo-surjectivity (mentioned in Section 2) of generators in EF in [7, 8]: to show that a p-time generator has this property is essentially equivalent to the unprovability of dWPHP for it in the theory (PV or S^1_2, resp.), cf. [7, Sec.6] and [8].

Acknowledgments: I thank Igor C. Oliveira (Warwick U.) and Jan Pich (Oxford U.) for discussions about the topic.

References

[1] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson, Pseudorandom generators in propositional proof complexity, *SIAM J. on Computing*, **34(1)**, (2004), pp.67-88.

[12]The problem in [6] asked, in fact, a more precise question about Σ^b_1-conservativity after adding dWPHP.

[2] E. Allender, Applications of Time-Bounded Kolmogorov Complexity in Complexity Theory, in: Kolmogorov Complexity and Computational Complexity, ed.O.Watanabe, Monographs in Theoretical Computer Science, EATCS Ser., Springer-Verlag, (1992), pp.4-22.

[3] S. A. Cook and R. A. Reckhow, The relative efficiency of propositional proof systems, *J. Symbolic Logic*, **44(1)**, (1979), pp.36-50.

[4] S.Hirara, Unexpected Hardness Results for Kolmogorov Complexity Under Uniform Reductions, in: Proc. of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC), June 2020, pp.1038-1051.

[5] J. Krajíček, *Bounded arithmetic, propositional logic, and complexity theory*, Encyclopedia of Mathematics and Its Applications, Vol. **60**, Cambridge University Press, (1995).

[6] J. Krajíček, On the weak pigeonhole principle, *Fundamenta Mathematicae*, Vol.**170(1-3)**, (2001), pp.123-140.

[7] J. Krajíček, Tautologies from pseudo-random generators, *Bulletin of Symbolic Logic*, **7(2)**, (2001), pp.197-212.

[8] J. Krajíček, Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds, *J. of Symbolic Logic*, **69(1)**, (2004), pp.265-286.

[9] J. Krajíček, Diagonalization in proof complexity, *Fundamenta Mathematicae*, **182**, (2004), pp.181-192.

[10] J. Krajíček, A proof complexity generator, in: *Proc. from the 13th Int. Congress of Logic, Methodology and Philosophy of Science (Beijing, August 2007)*, King's College Publications, London, ser. Studies in Logic and the Foundations of Mathematics. Eds. C.Glymour, W.Wang, and D.Westerstahl, (2009), pp.185-190.

[11] J. Krajíček, *Forcing with random variables and proof complexity*, London Mathematical Society Lecture Note Series, No. **382**, Cambridge University Press, (2011).

[12] J. Krajíček, On the proof complexity of the Nisan-Wigderson generator based on a hard $NP \cap coNP$ function, *J. of Mathematical Logic*, **11(1)**, (2011), pp.11-27.

[13] J. Krajíček, On the computational complexity of finding hard tautologies, *Bulletin of the London Mathematical Society*, **46(1)**, (2014), pp.111-125.

[14] J. Krajíček, *Proof complexity*, Encyclopedia of Mathematics and Its Applications, Vol. **170**, Cambridge University Press, (2019).

[15] J. Krajíček, Small circuits and dual weak PHP in the universal theory of p-time algorithms, *ACM Transactions on Computational Logic*, 22, 2, Article 11 (May 2021).

[16] J. Krajíček, Information in propositional proofs and algorithmic proof search, *J. of Symbolic Logic*, vol.87, nb.2, (June 2022), pp.852-869.

[17] A. A. Razborov, Pseudorandom generators hard for k-DNF resolution polynomial calculus resolution, *Annals of Mathematics*, **181(2)**, (2015), pp.415-472.

[18] H.Ren, R.Santhanam and Z.Wang, On the Range Avoidance Problem for Circuits, ECCC Report nb.48, (2022).