Electronic Colloquium on Computational Complexity, Report No. 137 (2007)

Lower Bounds for Kernelizations

Yijia Chen [*]
Shanghai Jiaotong University

Jörg Flum [†]
Albert-Ludwigs-Universität Freiburg

Moritz Müller [‡]
Albert-Ludwigs-Universität Freiburg

Abstract

Among others, refining the methods of [5] we improve a result of this paper and show for any parameterized problem with a "linear weak OR" and with NP-hard underlying classical problem that there is no polynomial reduction from the problem to itself that assigns to every instance x with parameter k an instance y with $|y| = k^{O(1)} \cdot |x|^{1-\varepsilon}$ unless the polynomial hierarchy collapses to its third level (here ε is any given real number greater than zero).

1. Introduction

Often, if a computationally hard problem must be solved in practice, one tries, in a preprocessing step, to reduce the size of the input data. This approach has been widely studied and applied in parameterized complexity and it is known as *kernelization* there. We recall the basic concepts.

Parameterized complexity is a refinement of classical complexity theory, in which one measures the complexity of an algorithm not only in terms of the total input length n, but also takes into account other aspects of the input codified as the parameter k. Central to parameterized complexity theory is the notion of fixed-parameter tractability. It relaxes the classical notion of tractability by allowing algorithms whose running time can be exponential but only in terms of the parameter. This is based on the idea to choose the parameter in such a way that it can be assumed to be small for the instances one is interested in. To be precise, a problem is said to be *fixed-parameter tractable* if it can be decided by an *fpt-algorithm*, that is, an algorithm whose running time is $f(k) \cdot p(n)$, where f is an arbitrary computable function and p a polynomial.

A *kernelization* of a parameterized problem is a polynomial time algorithm $\mathbb{K}$ that computes for every instance x of the problem an equivalent instance $\mathbb{K}(x)$ of a size bounded in terms of k (the parameter of the instance x). This suggests a new method for designing fpt-algorithms: To decide a given instance x, we compute the kernel $\mathbb{K}(x)$ and then decide if $\mathbb{K}(x)$ is a yes-instance by brute-force. The converse holds, too: Every fixed-parameter tractable problem has a kernelization. The proof of this fact is easy; however it gives only a "trivial" kernel with no algorithmic impact.

Besides efficient computability, an important quality of a good kernelization is *small kernel size*. The notion of polynomial kernelization is an abstract model for small kernel size. A kernelization $\mathbb{K}$ is *polynomial* if there is a polynomial p such that for all instances x (with parameter k), the size of $\mathbb{K}(x)$ is bounded by $p(k)$.

Polynomial kernelizations are known for many parameterized problems (compare [10]). However, till recently, besides artificial problems, only few natural problems were known to have *no* polynomial kernelization. This has changed, since Fortnow and Santhanam [5] showed that no problem "closed under OR" has a polynomial kernelization (unless the polynomial hierarchy collapses). They give various applications, in particular, they showed that the problem SAT parameterized by the number of propositional variables of the input formula has no polynomial kernelization. Further applications of the main result of [5] are given in Bodlaender et al. [3].

In this paper we refine the machinery developed by Fortnow and Santhanam to obtain better lower bounds. Applied to the SAT problem we show:

[*]Email: `yijia.chen@cs.sjtu.edu.cn`
[†]Email: `joerg.flum@math.uni-freiburg.de`
[‡]Email: `moritz.mueller@math.uni-freiburg.de`

ISSN 1433-8092

Assume that the polynomial hierarchy does not collapse. Then for every $\varepsilon > 0$ there is no polynomial time algorithm that for every instance α of SAT with k variables computes an equivalent instance α' with

$$|\alpha'| \leq k^{O(1)} \cdot |\alpha|^{1-\varepsilon}. \tag{1}$$

This result is a particular instance of a general theorem (compare Theorem 26 for the precise statement) that yields lower bounds of the type in (1) for every problem "closed under linear weak OR." For problems satisfying a weaker condition, namely only being "closed under weak OR," we still get quite good lower bounds; in case of SAT it would be:

$$|\alpha'| \leq k^{O(1)} \cdot |\alpha|^{o(1)}. \tag{2}$$

As already mentioned, concrete kernelizations yield algorithms for solving parameterized problems efficiently for small parameter values. Conceptually similar are compression algorithms, even though the intention is slightly different: the question is whether one can efficiently compress every "long" instance x of a problem Q with "a short witness" to a shorter equivalent instance x' of a problem Q' (here equivalent means that $x \in Q$ if and only if $x' \in Q'$). "Such compression enables to succinctly store instances until a future setting will allow solving them, either via a technological or algorithmic breakthrough or simply until enough time has elapsed" (see [8]). By suitably generalizing the notion of a kernelization of a parameterized problem to the notion of a kernelization from some parameterized problem to another one, Fortnow and Santhanam [5] introduce a framework which allows to deal with kernelizations and compressions at the same time (in [5] a different terminology is used). Nevertheless we stick to the traditional notion of kernelization as we mainly address problems of parameterized complexity.

More precisely, the content of the different sections is the following. After recalling some definitions and fixing our notation in Section 2, we consider and analyze some basic questions concerning kernelizations in Section 3. In particular, we shall see that "most" parameterized problems have a polynomial kernelization if and only if they are self-compressible.

A kernelization is *strong* if the parameter of $\mathbb{K}(x)$ is less than or equal to the parameter of x. It is known that every parameterized problem that has a kernelization already has a strong kernelization. In Section 4 we derive a general result (Corollary 11) that shows that parameterized problems satisfying certain conditions have no strong *polynomial* kernelizations. As an application we get that the problem SAT has no strong polynomial kernelization if P $\neq$ NP and no strong subexponential kernelization if the exponential time hypothesis (ETH) holds.

In Section 5 we recall the results of Fortnow and Santhanam [5] (and of Bodlaender et al. [3]) relevant in our context. Section 6 and Section 7 are devoted to the generalizations of these results of type (2) and of type (1), respectively, already mentioned above.

2. Preliminaries

The set of natural numbers (that is, nonnegative integers) is denoted by $\mathbb{N}$. For a natural number n let $[n] := \{1, \ldots, n\}$. By log n we mean $\lceil \log n \rceil$ if an integer is expected. For $n = 0$ the term log n is undefined. We trust the reader's common sense to interpret such terms reasonably.

We identify problems (or languages) with subsets Q of $\{0,1\}^*$. Clearly, as done mostly, we present concrete problems in a verbal, hence uncodified form or as a set of strings over an arbitrary finite alphabet. We use both P and PTIME to denote the class of problems Q such that $x \in Q$ is solvable in polynomial time.

A *reduction from a problem Q to a problem Q'* is a mapping $R : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$ we have $(x \in Q \iff R(x) \in Q')$. We write $R : Q \leq^p Q'$ if R is a reduction from Q to Q' computable in polynomial time, and $Q \leq^p Q'$ if there is a polynomial time reduction from Q to Q'.

2.1. Parameterized Complexity. A *parameterized problem* is a pair (Q, κ) consisting of a classical problem $Q \subseteq \{0,1\}^*$ and a *parameterization* $\kappa : \{0,1\}^* \to \mathbb{N}$, which is required to be polynomial time computable even if the result is encoded in unary.

We introduce some parameterized problems, which will be used later, thereby exemplifying our way to represent parameterized problems. We denote by p-SAT the parameterized problem

p-SAT	
Instance:	A propositional formula α in conjunctive normal form.
Parameter:	Number of variables of α.
Question:	Is α satisfiable?

By p-PATH and p-CLIQUE we denote the problems:

p-PATH	
Instance:	A graph G and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does G have a path of length k?

p-CLIQUE	
Instance:	A graph G and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does G have a clique of size k?

Similarly we define p-DOMINATING-SET.

We recall the definitions of the classes FPT, EXPT, EPT and SUBEPT. A parameterized problem (Q, κ) is *fixed-parameter tractable* (or, in FPT) if $x \in Q$ is solvable in time $f(\kappa(x)) \cdot |x|^{O(1)}$ for some computable $f : \mathbb{N} \to \mathbb{N}$. If f can be chosen such that $f(k) = 2^{\kappa(x)^{O(1)}}$, then (Q, κ) is in EXPT. If f can be chosen such that $f(k) = 2^{O(k)}$, then (Q, κ) is in EPT. If f can be chosen such that $f(k) = 2^{o^{\text{eff}}(k)}$, then (Q, κ) is in SUBEPT.

Here o^{eff} denotes the effective version of little oh: For computable functions $f, g : \mathbb{N} \to \mathbb{N}$ we say that f *is effectively little oh of* g and write $f = o^{\text{eff}}(g)$ if there is a *computable*, nondecreasing and unbounded function $\iota : \mathbb{N} \to \mathbb{N}$ such that for sufficiently large $k \in \mathbb{N}$

$$f(k) \leq \frac{g(k)}{\iota(k)}.$$

As usual we often write $f(k) = o^{\text{eff}}(g(k))$ instead of $f = o^{\text{eff}}(g)$.

At some places in this paper, it will be convenient to consider *preparameterized problems*; these are pairs (Q, κ), where again Q is a classical problem and κ is a *preparametrization*, that is, an arbitrary function from $\{0,1\}^*$ to the set $\mathbb{R}_{\geq 0}$ of nonnegative real numbers.

3. Kernelizations

In this section we start by recalling the notion of kernelization and by introducing some refinements. We study some basic properties of kernelizations and its relationship to the notion of compression.

Definition 1. Let (Q, κ) be a parameterized problem and $f : \mathbb{N} \to \mathbb{N}$ be a function. An *f-kernelization* for (Q, κ) is a polynomial time algorithm $\mathbb{K}$ that on input $x \in \{0,1\}^*$ outputs $\mathbb{K}(x) \in \{0,1\}^*$ such that

$$(x \in Q \iff \mathbb{K}(x) \in Q) \quad \text{and} \quad |\mathbb{K}(x)| \leq f(\kappa(x)).$$

In particular, $\mathbb{K}$ is a reduction from Q to itself. If in addition for all $x \in \{0,1\}^*$

$$\kappa(\mathbb{K}(x)) \leq \kappa(x),$$

then $\mathbb{K}$ is a *strong f*-kernelization.

We say that (Q, κ) has a *linear, polynomial, subexponential, simply exponential, and exponential kernelization* if there is an f-kernelization for (Q, κ) with $f(k) = O(k)$, $f(k) = k^{O(1)}$, $f(k) = 2^{o^{\text{eff}}(k)}$, $f(k) = 2^{O(k)}$, and $f(k) = 2^{k^{O(1)}}$, respectively.

The following result is well-known:

Proposition 2. *Let (Q, κ) be a parameterized problem. The following statements are equivalent.*

(1) (Q, κ) is fixed-parameter tractable.

(2) (Q, κ) has an f-kernelization for some computable f.

(3) (Q, κ) has a strong f-kernelization for some computable f.

The recent survey [7] contains examples of natural problems whose currently best known kernelizations are polynomial, simply exponential and exponential.

We are mainly interested in polynomial kernelizations. First we show that the notions of polynomial kernelization and of strong polynomial kernelization are distinct:

Proposition 3. *There is a parameterized problem (Q, κ) that has a polynomial kernelization but no strong polynomial kernelization.*

Proof: Let Q be a classical problem that is not solvable in time $2^{O(|x|)}$. We define a parameterized problem (P, κ) with $P \subseteq \{0,1\}^* \times \{1\}^*$ and with $\kappa((x, 1^k)) = k$. By 1^k we denote the string consisting of k many 1s. For each $k \in \mathbb{N}$ we define the *k-projection $P[k] := \{x \mid (x, 1^k) \in P\}$ of P* by:

– If $k = 2\ell + 1$, then
$$P[k] := Q_{=\ell} \; (:= \{x \in Q \mid |x| = \ell\}).$$
Hence, all elements in $P[k]$ have length ℓ.

– If $k = 2\ell$, then
$$P[k] := \big\{x1^{2^\ell} \mid x \in Q_{=\ell}\big\},$$
where $x1^{2^\ell}$ is the concatenation of x with the string 1^{2^ℓ}. Hence, all elements in $P[k]$ have length $\ell + 2^\ell$.

Intuitively, an element in the 2ℓ-projection is an element in the $(2\ell + 1)$-projection padded with 2^ℓ many 1s. It is not hard to see that P has a linear kernelization (which on the even projections increases the parameter).

We claim that P has no strong polynomial kernelization. Assume $\mathbb{K}$ is such a kernelization and $c \in \mathbb{N}$ such that
$$|\mathbb{K}((z, 1^m))| \le m^c.$$
We use $\mathbb{K}$ to solve $x \in Q$ in time $2^{O(|x|)}$:

Let x be an instance of Q and let $\ell := |x|$. We may assume that
$$(2\ell)^c < 2^\ell$$
(note that there are only finitely many x not satisfying this inequality). We compute (in time $2^{O(\ell)}$)
$$(u, k) := \mathbb{K}\big((x1^{2^\ell}, 2\ell)\big).$$
We know that $k \le 2\ell$ and $|u| \le (2\ell)^c < 2^\ell$. If u does not have the length of the strings in $P[k]$, then $(u, k) \notin P$ and therefore $x \notin Q$. In particular, this is the case if $k = 2\ell$ (as $|u| < 2^\ell$). If u has the length of the strings in $P[k]$ and hence $k < 2\ell$, then it is easy to read off from u an instance y with $|y| < |x|$ and ($y \in Q \iff x \in Q$). We then apply the same procedure to y. □

Remark 4. Let $c \in \mathbb{N}$. It is not hard to generalize the previous example and to show that there is a parameterized problem with a polynomial kernelization but with no polynomial kernelization $\mathbb{K}$ satisfying for all $x \in \{0,1\}^*$
$$\kappa(\mathbb{K}(x)) \le \kappa(x)^c.$$

The next result shows that a parameterized problem (Q, κ) in FPT$\setminus$EXPT with $Q \in$ NP cannot have polynomial kernelizations. We show a little bit more. Recall that EXP is the class of classical problems Q such that $x \in Q$ is solvable in deterministic time $2^{|x|^{O(1)}}$.

Proposition 5. *Assume that the problem (Q, κ) has a polynomial kernelization and that $Q \in$ EXP. Then $(Q, \kappa) \in$ EXPT.*

Proof: Let $\mathbb{K}$ be a polynomial kernelization of (Q, κ). As $Q \in$ EXP there is a deterministic algorithm $\mathbb{A}$ solving $x \in Q$ in time $2^{|x|^{O(1)}}$. The algorithm that on $x \in \{0,1\}^*$ first computes $\mathbb{K}(x)$ and then applies $\mathbb{A}$ to $\mathbb{K}(x)$ solves $x \in Q$ in time $|x|^{O(1)} + 2^{|\mathbb{K}(x)|^{O(1)}} = 2^{|\kappa(x)|^{O(1)}} \cdot |x|^{O(1)}$. $\square$

The model-checking of monadic second-order logic on the class of trees is in EXP. By a result of [6] the corresponding parameterized problem with the length of the formula as parameter is in FPT $\setminus$ EXPT unless P $=$ NP. Hence, by the preceding proposition, it has no polynomial kernelization (unless P $=$ NP).

In later sections, under some complexity-theoretic assumptions, we will present various examples of natural problems that are in EPT and have no polynomial kernelization. Here we give a simple, artificial example without polynomial kernelizations which holds unconditionally. Bodlaender et al. [3] even construct an example of a problem in EPT without subexponential kernelizations.

Example 6. Let Q be a classical problem not in PTIME but solvable in time $O(|x|^{\log|x|})$. Let κ be the parameterization mapping x to $(\log|x|)^2$. Then $(Q, \kappa) \in$ EPT, because $2^{\kappa(x)} = |x|^{\log|x|}$.

For the sake of contradiction assume that (Q, κ) has a polynomial kernelization $\mathbb{K}$. Then to decide if $x \in Q$ it suffices to decide if $\mathbb{K}(x) \in Q$. Since $|\mathbb{K}(x)| = (\log|x|)^{O(1)}$ this can be done in time

$$|\mathbb{K}(x)|^{\log|\mathbb{K}(x)|} \leq (\log|x|)^{O(\log\log|x|)} \leq 2^{(\log\log|x|)^{O(1)}} \leq |x|^{O(1)}.$$

Thus $Q \in$ PTIME, a contradiction.

Next we show that the different degrees of kernelizability introduced in Definition 1 are indeed different.

Proposition 7. *The classes of parameterized problems with a linear, a polynomial, a subexponential, a simply exponential, and an exponential kernelization are pairwise different.*

The claim immediately follows from the following lemma.

Lemma 8. *Let $g : \mathbb{N} \to \mathbb{N}$ be nondecreasing and unbounded and $f : \mathbb{N} \to \mathbb{N}$ such that $f(k) \leq g(k-1)$ for all sufficiently large k. Then there is a $Q \subseteq \{0,1\}^*$ and a preparameterization κ such that (Q, κ) has a g-kernelization but no f-kernelization.*

If in addition g is increasing and time-constructible, then we can choose κ to be a parameterization.

Proof: Let g and f be as in the statement. We choose k_0 such that $f(k) \leq g(k-1)$ for all $k \geq k_0$. We consider the "inverse function" ι_g of g given by

$$\iota_g(m) := \min\{s \in \mathbb{N} \mid g(s) \geq m\}.$$

Then for all $n \in \mathbb{N}$

$$n \leq g(\iota_g(n)) \qquad \text{and} \qquad \text{if } \iota_g(n) \geq 1, \text{then } \ g(\iota_g(n) - 1) < n. \tag{3}$$

Let Q be a problem not in PTIME and define the parameterization κ by $\kappa(x) := \iota_g(|x|)$. By the first inequality in (3) the identity is a g-kernelization of (Q, κ).

Assume that there is an f-kernelization $\mathbb{K}$ of (Q, κ). As ι_g is unbounded, we have $\iota_g(|x|) \geq k_0$ for sufficiently long $x \in \{0,1\}^*$. Then

$$|\mathbb{K}(x)| \leq f(\kappa(x)) = f(\iota_g(|x|)) \leq g(\iota_g(|x|) - 1) < |x|.$$

Thus applying $\mathbb{K}$ at most $|x|$ times we get an equivalent instance of length at most $f(k_0)$. Therefore, $Q \in$ PTIME, a contradiction.

If g is increasing and time-constructible, then ι_g is polynomial time computable and hence κ is a parameterization. $\square$

Polynomial Kernelization and Compression. Most natural problems $Q \in$ NP have a *canonical* representation of the form

$$x \in Q \quad \iff \quad \text{there is } y \in \{0,1\}^{g(x)} \text{ such that } (x,y) \in Q_0 \tag{4}$$

for some polynomial time computable function $g : \{0,1\}^* \to \mathbb{N}$ and some $Q_0 \in$ PTIME. In [2] the problem (Q, g) has been called the *canonical parameterization* of Q (more precisely, one should speak of the canonical parameterization induced by the representation (4) of Q). Clearly (Q, g) is fixed-parameter tractable, it is even in EPT. If (Q, κ) was a parameterized problem, then (Q, g) is called the *canonical reparameterization* of (Q, κ).

The canonical reparameterization of p-SAT is p-SAT itself; the canonical reparameterizations of the problems p-PATH, p-CLIQUE and p-DOMINATING-SET are the problems *uni*-PATH, *uni*-CLIQUE and *uni*-DOMINATING-SET, respectively, where in the three cases, we have $g((G, k)) = k \cdot \log |V|$; hence in particular,

uni-PATH	
Instance:	A graph $G = (V, E)$ and $k \in \mathbb{N}$.
Parameter:	$k \cdot \log |V|$.
Question:	Does G have a path of length k?

Many fixed-parameter tractable problems, namely all in EXPT and hence, in particular, p-PATH, have a polynomial kernelization if and only if their canonical reparameterizations have. This is shown by the following proposition.

Proposition 9. *Let $(Q, \kappa) \in$ EXPT and let (Q, g) be the canonical reparameterization of (Q, κ). Assume that g has the form*

$$g(x) = \kappa(x) \cdot \log h(x) \text{ with } h(x) = |x|^{O(1)}$$

and $h(x) \geq 2$ for sufficiently large x. Then

(Q, κ) has a polynomial kernelization iff (Q, g) has a polynomial kernelization.

Proof: Clearly, every polynomial kernelization of (Q, κ) is a polynomial kernelization of (Q, g). Conversely, let $\mathbb{K}$ be a polynomial kernelization of (Q, g). Choose $c, c' \in \mathbb{N}$ and an algorithm $\mathbb{A}$ solving $x \in Q$ in time $2^{\kappa(x)^c}|x|^{c'}$. We define a polynomial kernelization $\mathbb{K}'$ for (Q, κ).

Fix $x_+ \in Q$ and $x_- \notin Q$. (If Q is trivial, that is, $Q = \emptyset$ or $Q = \{0,1\}^*$, we let $\mathbb{K}'$ always output the empty string.) Let $x \in \{0,1\}^*$. If $\kappa(x) < (\log |x|)^{1/c}$, the algorithm $\mathbb{A}$ on input x needs at most $|x|^{c'+1}$ steps. In this case we let $\mathbb{K}'(x)$ be x_+ or x_- according to the answer of $\mathbb{A}$. Otherwise $\kappa(x)^c \geq \log |x|$. Then $|\mathbb{K}(x)| = (\kappa(x) \cdot \log h(x))^{O(1)} = (\kappa(x) \cdot \log |x|)^{O(1)} = \kappa(x)^{O(1)}$, so we can set $\mathbb{K}'(x) := \mathbb{K}(x)$. $\Box$

The reader familiar with the paper of [8] will realize that this result shows that any parameterized problem (Q, κ) in EXPT has a polynomial kernelization if and only if the problem Q is self-compressible.

4. Excluding strong kernelizations

In this section we exemplify how self-reducibility can be used to rule out *strong* polynomial kernelizations. This method is very simple and works under the assumption that P $\neq$ NP. We use it to give two natural examples of problems in EPT that do not have *strong* polynomial kernelizations.

We will revisit these examples in section 5. There we will see that these problems do not even have polynomial kernelizations using the stronger assumption that the polynomial hierarchy does not collapse to its third level.

Lemma 10. *Let (Q, κ) be a parameterized problem and assume that the 0th slice $Q(0) := \{x \in Q \mid \kappa(x) = 0\}$ is in* PTIME. *If there is a polynomial (subexponential) kernelization $\mathbb{K}$ such that for all $x \notin Q(0)$*

$$\kappa(\mathbb{K}(x)) < \kappa(x), \tag{5}$$

then $Q \in$ PTIME $\quad((Q, \kappa) \in$ SUBEPT).

Proof: Let $\mathbb{K}$ be a kernelization satisfying (5). The following algorithm $\mathbb{A}$ decides Q (using a polynomial time decision procedure $\mathbb{B}$ for $Q(0)$). Given an instance x of Q, the algorithm $\mathbb{A}$ computes $\mathbb{K}(x), \mathbb{K}(\mathbb{K}(x)), \ldots$; by (5) after at most $\kappa(x)$ steps we obtain an instance y with $\kappa(y) = 0$; hence $(x \in Q \iff y \in Q(0))$; now $\mathbb{A}$ simulates $\mathbb{B}$ on y.

If $\mathbb{K}$ was a polynomial kernelization, say, $|\mathbb{K}(x)| \leq \kappa(x)^c$, then, again by (5), all of $|\mathbb{K}(\mathbb{K}(x))|, |\mathbb{K}(\mathbb{K}(\mathbb{K}(x)))|, \ldots$ are bounded by $\kappa(x)^c$. Recall that parameterizations are computable in polynomial time even if the result is encoded in unary. Hence $\kappa(x) = |x|^{O(1)}$. It follows that $\mathbb{A}$ runs in polynomial time.

If $\mathbb{K}$ was a subexponential kernelization, say, $|\mathbb{K}(x)| \leq 2^{\kappa(x)/\iota(\kappa(x))}$ with computable, nondecreasing and unbounded ι and $\mathbb{K}(x)$ is computable in time $|x|^d$, then $\mathbb{A}$ needs to compute the equivalent instance y at most

$$|x|^d + 2^{d \cdot \kappa(x)/\iota(\kappa(x))} + 2^{d \cdot (\kappa(x)-1)/\iota(\kappa(x)-1)} + 2^{d \cdot (\kappa(x)-2)/\iota(\kappa(x)-2)} + \ldots + 2^{d \cdot 1/\iota(1)},$$

many steps. As we can assume that the function $j \mapsto j/\iota(j)$ is increasing, this number of steps is bounded by $|x|^d + \kappa(x) \cdot 2^{d \cdot \kappa(x)/\iota(\kappa(x))}$, which shows that $(Q, \kappa) \in$ SUBEPT. □

Corollary 11. *Let (Q, κ) be a parameterized problem with $Q(0) \in$ PTIME. Assume that there is a polynomial reduction R from Q to itself which is* parameter decreasing, *that is, for all $x \notin Q(0)$,*

$$\kappa(R(x)) < \kappa(x).$$

– *If (Q, κ) has a strong polynomial kernelization, then $Q \in$ PTIME.*

– *If (Q, κ) has a strong subexponential kernelization, then $(Q, \kappa) \in$ SUBEPT.*

Proof: Let R be as in the statement and let $\mathbb{K}$ be a strong polynomial (subexponential) kernelization of (Q, κ). Then the composition $\mathbb{K} \circ R$, that is, the mapping $x \mapsto \mathbb{K}(R(x))$, is a polynomial (subexponential) kernelization of (Q, κ) satisfying (5); hence, by the previous lemma, we get $Q \in$ PTIME ($Q \in$ SUBEPT). □

Examples 12. The classical problems underlying

$$p\text{-SAT} \quad \text{and} \quad p\text{-POINTED-PATH}$$

have parameter-decreasing polynomial reductions to themselves, where

p-POINTED-PATH	
Instance:	A graph $G = (V, E)$, a vertex $v \in V$, and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does G have a path of length k starting at v?

Proof: p-SAT: We define a parameter-decreasing polynomial reduction R from p-SAT to itself as follows: Let α be a CNF formula. If α has no variables, we set $R(\alpha) := \alpha$. Otherwise let X be the first variable in α. We let $R(\alpha)$ be a formula in CNF equivalent to

$$(\alpha\frac{\text{TRUE}}{X} \vee \alpha\frac{\text{FALSE}}{X}),$$

where, for example, $\alpha\frac{\text{TRUE}}{X}$ is the formula obtained from α by replacing X by TRUE everywhere. Clearly $R(\alpha)$ can be computed from α in polynomial time.

p-POINTED-PATH: We define a parameter-decreasing polynomial reduction R from p-POINTED-PATH to itself as follows: Let (G, v, k) be an instance of p-POINTED-PATH and assume $k \geq 3$. For any path $P : v, v_1(P), v_2(P)$ of length 2 starting from v let G_P be the graph obtained from G by deleting the two vertices $v, v_1(P)$ (and all the edges incident with one of these vertices). Let H be the graph obtained from the disjoint union of all the graphs G_P (where P ranges over all paths of length 2 starting in v) by adding a new vertex w and all edges $\{w, v_2(P)\}$. Then H has a path of length $(k-1)$ starting at w if and only if G has a path of length k starting at v. Hence we can set $R((G, v, k)) := (H, w, k-1)$. □

Corollary 13. (1) If P $\neq$ NP, then p-SAT has no strong polynomial kernelization.

(2) If ETH holds, then p-SAT has no strong subexponential kernelization.

(3) If P $\neq$ NP, then p-POINTED-PATH has no strong polynomial kernelization.

(4) If ETH holds, then p-POINTED-PATH has no strong subexponential kernelization.

Proof: Part (1) and (3) are immediate by Corollary 11. Moreover, we know by this corollary that if one of the two problems has a strong subexponential kernelization, then it is in SUBEPT. However then ETH would fail in the case of p-SAT by [9] and in the case of p-POINTED-PATH by [1]. □

5. Excluding polynomial kernelizations

The following type of reductions that preserve polynomial kernels was introduced in [5] (based on a notion of [8]) under the name "W-reductions."

Definition 14. Let (Q, κ) and (Q', κ') be parameterized problems. A *polynomial reduction* from (Q, κ) to (Q', κ') is a polynomial reduction R from Q to Q' such that

$$\kappa'(R(x)) = \kappa(x)^{O(1)}.$$

We then write $R : (Q, \kappa) \leq^p (Q', \kappa')$. Furthermore $(Q, \kappa) \leq^p (Q', \kappa')$ means that there is a polynomial reduction from (Q, κ) to (Q', κ').

Example 15. *uni*-PATH $\leq^p$ p-SAT.

Proof: Let (G, k) with $G = (V, E)$ be an instance of *uni*-PATH. We may assume that $V = [0, n-1]$ and (by adding isolated points if necessary) that n is a power of 2. We will assign to (G, k) a formula α in CNF containing variables $X_{s,i}$ with $s \in [\log\, n]$ and $i \in [k]$ with the intended meaning "the sth bit of the ith vertex of a path of length k is 1." For $i, j \in [k]$, $i \neq j$ one has to express by a clause that the selected vertices as ith and jth point of the path are distinct and for $i \in [k-1]$ that the ith and the $(i+1)$th selected vertices are related by an edge. For example the second one may be expressed by letting be for every $i \in [k-1]$ and every $u, v \in V$ with $\{u, v\} \notin E$

$$\bigvee_{s \in [\log\, n]} \neg X_{s,i}^{\text{bit}(s,u)} \vee \bigvee_{s \in [\log\, n]} \neg X_{s,i+1}^{\text{bit}(s,v)},$$

a clause of α, where bit(s, u) denotes the sth bit in the binary representation of u of length log n and where $X^1 := X$ and $X^0 := \neg X$ for every variable X.

Then G has a path of length k if and only if α is satisfiable. As α has $k \cdot \log\, |V|$ variables, the mapping $(G, k) \mapsto \alpha$ is a polynomial reduction. □

Example 16 ([8]). p-SAT $\leq^p$ *uni*-DOMINATING-SET.

Polynomial reductions preserve polynomial kernelizations in the following sense:

Lemma 17. *Let (Q, κ) and (Q', κ') be parameterized problems. with*

$$(Q, \kappa) \leq^p (Q', \kappa') \quad \textit{and} \quad Q' \leq^p Q.$$

If (Q', κ') has a polynomial kernelization, then (Q, κ) has a polynomial kernelization.

Note that $Q' \leq^p Q$ is always satisfied for NP-complete problems Q and Q'.

Proof of Lemma 17: Let $R : (Q, \kappa) \leq^p (Q', \kappa')$ and $S : Q' \leq^p Q$. Assume that $\mathbb{K}$ is a polynomial kernelization for (Q', κ'). Then $S \circ \mathbb{K} \circ R$ is a polynomial kernelization for (Q, κ), as for all $x \in \{0, 1\}^*$

$$|S(\mathbb{K}(R(x)))| = |\mathbb{K}(R(x))|^{O(1)} = \kappa'(R(x))^{O(1)} = \kappa(x)^{O(1)}.$$

□

In order to exclude polynomial kernelizations using the previous lemma one needs a primal problem without a polynomial kernelization. The key to obtain such problems was found by Fortnow and Santhanam [5]. It is contained in the following theorem. It has been applied in [5, 3].

Definition 18. Let $Q, Q' \subseteq \{0, 1\}^*$ be classical problems. A *distillation from Q in Q'* is a polynomial time algorithm $\mathbb{D}$ that receives as inputs finite sequences $\bar{x} = (x_1, \ldots, x_t)$ with $x_i \in \{0, 1\}^*$ for $i \in [t]$ and outputs a string $\mathbb{D}(\bar{x}) \in \{0, 1\}^*$ such that

(1) $|\mathbb{D}(\bar{x})| = \left(\max_{i \in [t]} |x_i|\right)^{O(1)}$;

(2) $\mathbb{D}(\bar{x}) \in Q'$ if and only if for some $i \in [t]$: $x_i \in Q$.

If $Q' = Q$ we speak of a *self-distillation*. We say that Q *has a distillation* if there is a distillation from Q in Q' for some Q'.

Theorem 19 (Fortnow and Santhanam [5]). *No* NP*-hard problem has a distillation unless* PH $= \Sigma_3^{\text{P}}$ *(that is, unless the polynomial hierarchy* PH *collapses to its third level* Σ_3^{P}*).*

To see how this result (and the polynomial reductions) can be used to exclude polynomial kernelizations we include applications from [3] and [5].

Corollary 20 ([3]). *p-*PATH *has no polynomial kernelization unless* PH $= \Sigma_3^{\rm P}$.

Proof: We assume that p-PATH has a polynomial kernelization $\mathbb{K}$ and show that then the (classical) problem PATH has a self-distillation to itself. In fact, let $(G_1, k_1), \dots, (G_t, k_t)$ be instances of PATH. We assume that all $k_i = k$ for some k (if this is not the case let $k := 1 + 2 \cdot \max_{i\in[t]} k_i$ and add to every G_i a path of length $k - k_i - 1$ with one endpoint connected to all vertices of G_i). Let G be the disjoint union of all the graphs G_i. Clearly, G has a path of length k if and only if there exists an $i \in [t]$ such that G_i has a path of length k. As $|\mathbb{K}((G, k))|$ is polynomially bounded in k and hence in $\max_{i\in[t]} \|(G_i, k_i)\|$, the mapping $(G_1, k_1), \dots, (G_t, k_t) \mapsto \mathbb{K}((G, k))$ is a self-distillation of PATH. □

Corollary 21 ([5]). *The problems*

p-SAT *and uni*-DOMINATING-SET

have no polynomial kernelization unless PH $= \Sigma_3^{\rm P}$.

Proof: Assume PH $\neq \Sigma_3^{\rm P}$. By the previous corollary we know that p-PATH has no polynomial kernelization. Hence, as p-PATH $\in$ EPT, its canonical reparametrization *uni*-PATH has no polynomial kernelization by Proposition 9. The remaining claims follow from examples 15 and 16 by Lemma 17. □

We know that no NP-hard problem has a self-distillation (unless PH $= \Sigma_3^{\rm P}$). Clearly each problem in PTIME has a self-distillation.

Proposition 22. *If* NE $\neq$ E*, then there is a problem in* NP $\setminus$ P *that has a self-distillation.*

By E and NE we denote the class of problems Q such that $x \in Q$ is solvable by a deterministic algorithm and a nondeterministic algorithm, respectively, in time $2^{O(|x|)}$.

Proof of Proposition 22: Let $Q_0 \subseteq \{0, 1\}^*$ be a language in NE $\setminus$ E. We assume that each yes instance of Q_0 starts with a 1, and can thus be viewed as a natural number in binary. For $n \in \mathbb{N}$ let *bin*(n) denote its binary representation. We set

$$Q := \{1^n \mid \textit{bin}(n) \in Q_0\}\,.$$

It is easy to see that $Q \in$ NP $\setminus$ P. Now let Q' be the "OR-closure" of Q, that is

$$Q' := \{(x_1, \dots, x_m) \mid m \geq 1 \text{ and } x_i \in Q \text{ for some } i \in [m]\}\,.$$

Again it is easy to see that $Q' \in$ NP $\setminus$ P. We claim that Q' has a self-distillation.

Let $(x_{11}, \dots, x_{1m_1}), \dots, (x_{t1}, \dots, x_{tm_t})$ be a sequence of instances of Q'. We can assume that all x_{ij} are sequences of 1s (otherwise we simply ignore those which are not). Let n be the maximal length of the x_{ij}. Then

$$\{x_{11}, \dots, x_{1m_1}, \dots, x_{t1}, \dots, x_{tm_t}\} = \{y_1, \dots, y_q\}$$

for some $q \leq n$. Thus $(y_1, \dots, y_q)$ has length $O(n^2)$. Clearly $(y_1, \dots, y_q)$ is in Q' if and only if $(x_{i1}, \dots, x_{im_i}) \in Q'$ for some $i \in [t]$. □

6. Strong lower bounds

In this section and the next one, by a careful analysis of the proof of Theorem 19, we obtain improvements, which yield better lower bounds for kernelizations. In particular for the path problem we will show:

Theorem 23. *Let $\varepsilon > 0$ and assume* PH $\neq \Sigma_3^{\rm P}$*. Then there is* no *polynomial reduction from* PATH *to itself computing for each instance (G, k) of* PATH *an instance (G', k') with*

$$\|G'\| = k^{O(1)} \cdot \|G\|^{1-\varepsilon}.$$

This result will be a special instance of a more general result stating similar lower bounds for problems sharing the following property:

Definition 24. Let (Q, κ) be a parameterized problem. A *linear weak OR for (Q, κ)* is a polynomial time algorithm $\mathbb{O}$ that for every finite tuple $\bar{x} = (x_1, \dots, x_t)$ of instances of Q outputs an instance $\mathbb{O}(\bar{x})$ of Q such that

(1) $|\mathbb{O}(\bar{x})| = t \cdot \big(\max_{i\in[t]}|x_i|\big)^{O(1)}$;

(2) $\kappa(\mathbb{O}(\bar{x})) = \big(\max_{i\in[t]}|x_i|\big)^{O(1)}$;

(3) $\mathbb{O}(\bar{x}) \in Q$ if and only if for some $i \in [t]$: $x_i \in Q$.

Examples 25. (a) The parameterized problem p-PATH has a linear weak OR. This can be seen using the construction in Corollary 20.

(b) The parameterized problem p-SAT has a linear weak OR.

Proof: We define a linear weak OR $\mathbb{O}$. Let $\alpha_1, \dots, \alpha_t$ be CNF formulas, say, α_i a formula with n_i variables. We set

$$n := \max_{i\in[t]} n_i \quad \text{and} \quad m := \max_{i\in[t]}|\alpha_i|.$$

We may assume that all α_i have variables in $\{X_1, \dots, X_n\}$ and that $\log t$ is a natural number (if t is not a power of two we duplicate one of the formulas for an appropriate number of times).

If $t \geq 2^n$, the algorithm $\mathbb{O}$ proves whether one of the α_is is satisfiable (by systematically checking all assignments) and outputs a CNF formula $\mathbb{O}(\alpha_1, \dots, \alpha_t)$ satisfying condition (3) of the preceding definition.

Assume $t < 2^n$. We introduce $\log t$ new variables $Y_1, \dots, Y_{\log t}$. For $i \in [t]$ we set

$$\beta_i := \bigwedge_{s\in[\log t]} Y_s^{\text{bit}(s,i)}$$

(recall that $\text{bit}(s,i)$ denotes the sth bit in the binary representation of i and that $X^1 = X$ and $X^0 = \neg X$ for every variable X).

We bring each $(\beta_i \to \alpha_i)$ into conjunctive normal form: Assume $\alpha_i = \bigwedge_\ell \bigvee_{\ell'} \lambda_{\ell\ell'}$ with literals $\lambda_{\ell\ell'}$, then $(\beta_i \to \alpha_i)$ is equivalent to

$$\gamma_i := \bigwedge_\ell \Big(\bigvee_{s\in[\log t]} Y_s^{1-\text{bit}(s,i)} \vee \bigvee_{\ell'} \lambda_{\ell\ell'}\Big).$$

We let γ be the CNF formula $\gamma := \bigwedge_{i\in[t]} \gamma_i$. We set $\mathbb{O}(\alpha_1, \dots, \alpha_t) := \gamma$.

Clearly $\mathbb{O}$ is computable in polynomial time. Furthermore, by construction the formula $\mathbb{O}(\alpha_1, \dots, \alpha_t)$ is equivalent to $\bigwedge_{i\in[t]}(\beta_i \to \alpha_i)$. Because any assignment to $Y_1, \dots, Y_{\log t}$ satisfies exactly one of the β_is, the formula $\mathbb{O}(\alpha_1, \dots, \alpha_t)$ is satisfiable if and only if there is an $i \in [t]$ such that α_i is satisfiable; hence condition (3) of Definition 24 is satisfied . Furthermore, $\mathbb{O}$ also satisfies the conditions (1) and (2). For (2) note that γ has $n + \log t$ variables. By our assumption on t, we have $n+\log t \leq 2n \leq 2m$. For (1) note that each γ_i has length $O(m\cdot(m+\log t))$ and hence, $\mathbb{O}(\alpha_1, \dots, \alpha_t)$ has length $O(m^3)$. □

(c) The parameterized problem *uni*-CLIQUE has a linear weak OR.

Proof: Let $(G_1, k_1), \dots, (G_t, k_t)$ be instances of *uni*-CLIQUE. Of course, we can assume that $k_i \leq |V_i|$, where V_i is the set of vertices of G_i. Let $k := \max_{i\in[t]} k_i$. By adding a clique of $k - k_i$ new vertices to G_i and connecting all new vertices to all old vertices in V_i we can pass to an instance (G'_i, k) equivalent to (G_i, k_i). Let $m := \max_{i\in[t]}|V'_i|$ $(\leq 2 \cdot \max_{i\in[t]}|V_i|)$.

If $t \geq 2^m$, by exhaustive search the algorithm $\mathbb{O}$ checks whether one of the G'_is has a clique of size k; if this is the case $\mathbb{O}$ outputs (G_i, k_i) for such a G'_i and otherwise it outputs, say, (G_1, k_1).

Assume that $t < 2^m$. We set $\mathbb{O}((G_1, k_1), \dots, (G_t, k_t)) := (G, k)$, where G denotes the disjoint union of the graphs G'_i. Clearly, $\mathbb{O}$ is computable in polynomial time and condition (3) is satisfied. For condition (1) note that we have for the set V of vertices of G the inequality $|V| \leq t \cdot m$. The parameter of $\mathbb{O}((G_1, k_1), \dots, (G_t, k_t))$ is $k \cdot \log |V| \leq k \cdot \log (t \cdot m) \leq k \cdot (m + \log m) = O(m^2)$. □

(d) The parameterized problem *uni*-DOMINATING-SET has a linear weak OR.

Proof: Let $(G_1, k_1), \dots, (G_t, k_t)$ be instances of *uni*-DOMINATING-SET. Let $k := \max_{i\in[t]} k_i$. By adding $k - k_i$ isolated vertices, we can pass to equivalent instances $(G'_1, k), \dots, (G'_t, k)$. Let $G'_i = (V'_i, E'_i)$. We may assume that $t > k$ and that the vertex sets V'_i are pairwise disjoint.

If $t \geq 2^m$, where $m := \max_{i\in[t]}|V'_i|$, the algorithm $\mathbb{O}$ checks by exhaustive search whether one of the G'_is has a dominating set of size k; if so $\mathbb{O}$ outputs (G_i, k_i) for such a G'_i and otherwise it outputs (G_1, k_1).

Assume that $t < 2^m$. For $i \in [t]$ and $j \in [0,k] := \{0, 1, \ldots, k\}$ let $V_i'(j)$ be a copy of V_i', say,

$$V_i'(j) := \{(v, j) \mid v \in V_i'\}.$$

Let $G = (V, E)$ be the graph with vertex set

$$V := \bigcup_{s \in [\log\, t]} \{s(-), s(0), s(1)\} \;\cup \bigcup_{i \in [t], j \in [0,k]} V_i'(j)$$

The edge set E contains

- edges that make $\{s(-), s(0), s(1)\}$ a clique for $s \in [\log\, t]$;
- for $s \in [\log\, t]$ and $i \in [t]$ edges from $s(1)$ to all vertices in $V_i'(0)$ if $\text{bit}(s,i) = 0$ and edges from $s(0)$ to all vertices in $V_i'(0)$ if $\text{bit}(s,i) = 1$;
- for $i, i' \in [t]$, $v \in V_i',\ w \in V_{i'}'$, and $j, j' \in [0,k]$ the edge $\{(v,j), (w,j')\}$ if
  - $i \neq i'$ and $j = j' > 0$ or
  - $i = i'$ and $\{v, w\} \in E_i$ or
  - $i = i',\ j \neq j'$ and $v = w$.

We claim that

$$(G, k + \log\, t) \in \textit{uni}\text{-DOMINATING-SET} \iff \text{there is an } i \in [t]\text{: } (G_i', k) \in \textit{uni}\text{-DOMINATING-SET}. \tag{6}$$

For the backward direction assume for $i \in [t]$ that $\{v_1, \ldots, v_k\}$ is a dominating set in G_i'. Then

$$\{(v_1, 1), \ldots, (v_k, k)\} \cup \{s(\text{bit}(s,i)) \mid s \in [\log\, t]\}$$

is a dominating set of G.

For the forward direction let X be a dominating set of G of size $k + \log\, t$. For $s \in [\log\, t]$ in order to dominate the point $s(-)$ we see that at least one point of the clique $\{s(-), s(0), s(1)\}$ has to be contained in X.

Clearly, as $k < t$, there is an $i_0 \in [t]$ such that

$$X \cap \bigcup_{j \in [0,k]} V_{i_0}'(j) = \emptyset.$$

For $j \in [k]$ (in particular $j \neq 0$), in order to dominate the elements of $V_{i_0}'(j)$, the set X must contain an element of the form (v_j, j) with $v_j \in V_{i_j}'$ for some $i_j \neq i_0$. Moreover, as X only contains $k + \log\, t$ elements, the vertex v_j (and hence i_j) are uniquely determined by j. Then it is not hard to see that the set $\{v_j \mid j \in [k] \text{ and } i_j = i_1\}$ is a dominating set in G_{i_1}'. This finishes the proof of the equivalence (6).

We set $\mathbb{O}((G_1, k_1), \ldots, (G_t, k_t)) := (G, k)$. That $\mathbb{O}$ also satisfies condition (2) of a linear weak OR is shown as in the case of *uni*-CLIQUE. □

(e) The problem *alpha*-LCS has a linear weak OR. Here *alpha*-LCS denotes the canonical parameterization of the longest common subsequence problem:

alpha-LCS	
Instance:	An alphabet Σ, strings $X_1, \ldots, X_\ell \in \Sigma^*$, and $m \in \mathbb{N}$.
Parameter:	$m \cdot \log |\Sigma|$.
Question:	Is there a common subsequence of $X_1, \ldots, X_\ell$ of length m?

Proof: Let $(\Sigma_1, X_{11}, \ldots, X_{1\ell_1}, m_1) \ldots (\Sigma_t, X_{t1}, \ldots, X_{t\ell_t}, m_t)$ be instances of *alpha*-LCS. We can assume that $\ell_1 = \cdots = \ell_t = \ell$ (by repeating a sequence if necessary) and that $m_1 = \cdots = m_t = m$ (by adding $c_i^{m-m_i}$ to each X_{ij} for some new letter c_i). Moreover we can assume that the alphabets Σ_i are disjoint. Now we consider the ℓ strings over $\Sigma_1 \cup \ldots \cup \Sigma_t$

$$X_{11}X_{21}\ldots X_{t1}, \quad X_{12}X_{22}\ldots X_{t2}, \quad \ldots \quad X_{1\ell}X_{2\ell}\ldots X_{t\ell}$$

and the string $X_{t1}X_{(t-1)1}\dots X_{11}$.

One easily verifies that these $(\ell+1)$ strings have a common subsequence of length m if and only if for some $i \in [t]$ the strings $X_{i1},\dots,X_{i\ell_i}$ have one (for the forward direction note that a common subsequence of $X_{11}X_{21}\dots X_{t1}$ and $X_{t1}X_{(t-1)1}\dots X_{11}$ is a sequence over Σ_i for some $i \in [t]$). Now, if $t \geq \max_{i\in[t]}|\Sigma_i|^m$ we determine the value of $\mathbb{O}$ by exhaustive search and otherwise, we use the set of strings just constructed. □

Even though we could add further examples of parameterized problems with a linear weak OR, there are also many problems where we do not know whether they have a linear weak OR. We just mention one example, the problem *uni*-RED/BLUE-NONBLOCKER, the canonical reparametrization of the problem p-RED/BLUE-NONBLOCKER.

As we have seen that p-PATH has a linear weak OR, Theorem 23 follows from:

Theorem 26. *Let $\varepsilon > 0$. Let (Q,κ) be a parameterized problem with a linear weak OR and with NP-hard Q. Unless* $\mathrm{PH} = \Sigma_3^{\mathrm{P}}$, *there is* no *polynomial reduction from Q to itself that assigns to every instance x of Q an instance y with*

$$|y| = \kappa(x)^{O(1)} \cdot |x|^{1-\varepsilon}.$$

As we have seen that p-PATH has a linear weak OR, Theorem 23 is a special instance of Theorem 26. It will be convenient to reformulate Theorem 26. For this purpose we need some further notions.

Definition 27. A function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *pseudo-linear* if there is some $c \in \mathbb{N}$ and some $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$ such that for all $t \in \mathbb{N}$

$$f(t) \leq c \cdot t^{1-\varepsilon}.$$

The property that we need of pseudo-linear functions is contained in the following lemma. It is easy to prove.

Lemma 28. *Let $\varepsilon > 0$ and $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a pseudo-linear function. Then for every $c \in \mathbb{N}$ there exists a $d \in \mathbb{N}$ such that for sufficiently large n we have*

$$f(n^d) \cdot n^c + 1 \leq n^d.$$

Remark 29. As f will determine the lower bound stated in Theorem 26, it is worthwhile to note that a weak converse of the above lemma holds: Let f satisfy the conclusion of Lemma 28. Then there is some $\varepsilon > 0$ such that $f(t) < t^{1-\varepsilon}$ for infinitely many t.

To see this write $f(t) = t^{g(t)}$ for some g. Then for $c = 1$ there are $d, n_0 \in \mathbb{N}$ such that $n^{d\cdot g(n^d)} < n^{d-1}$ for all $n \geq n_0$. Thus $g(t) < 1 - 1/d$, i.e. $f(t) \leq t^{1-1/d}$, for $t = n_0^d, (n_0+1)^d, (n_0+2)^d \dots$. ⊣

For a parameterized problem (Q,κ), a constant $c \in \mathbb{N}$, and a function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ consider the preparameterized problem

$(Q, \kappa^c \times f)$	
Instance:	$x \in \{0,1\}^*$.
Parameter:	$\kappa(x)^c \cdot f(\lvert x\rvert)$.
Question:	$x \in Q$?

Theorem 26 follows from:

Lemma 30. *Let $c \in \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be pseudo-linear. Let (Q,κ) be a parameterized problem with a linear weak OR and with NP-hard Q. Then $(Q, \kappa^c \times f)$ has no linear kernelization, unless* $\mathrm{PH} = \Sigma_3^{\mathrm{P}}$.

We prove this lemma by generalizing Theorem 19.

Definition 31. Let $Q, Q' \subseteq \{0,1\}^*$ be classical problems and let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a function. A *linear f-distillation from Q in Q'* is a polynomial time algorithm $\mathbb{D}$ that receives as inputs finite sequences $\bar{x} = (x_1,\dots,x_t)$ with $x_i \in \{0,1\}^*$ for $i \in [t]$ and outputs a string $\mathbb{D}(\bar{x}) \in \{0,1\}^*$ such that

(1) $|\mathbb{D}(\bar{x})| = f(t) \cdot (\max_{i\in[t]}|x_i|)^{O(1)}$;

(2) $\mathbb{D}(\bar{x}) \in Q'$ if and only if for some $i \in [t]$: $x_i \in Q$.

We say that Q *has a linear f-distillation* if there is a linear f-distillation from Q in Q' for some problem Q'.

Lemma 32. *Let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be pseudo-linear. No NP-hard problem has a linear f-distillation unless* $\mathrm{PH} = \Sigma_3^{\mathrm{P}}$.

Proof: Let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be pseudo-linear and $Q \subseteq \{0,1\}^*$ be NP-hard. Assume that $\mathbb{D}$ is an f-distillation from Q in some problem Q'. We choose a constant $c \in \mathbb{N}$ such that

$$|\mathbb{D}(\bar{x})| \leq f(t) \cdot \Big(\max_{i \in [t]} |x_i|\Big)^c \tag{7}$$

for all $t \in \mathbb{N}$ and all sequences $\bar{x}$ of t instances of Q.

Let $\overline{Q} := \{0,1\}^* \setminus Q$ be the complement of Q and similarly $\overline{Q'}$ the complement of Q'. Clearly $\overline{Q}$ is coNP-hard. We show that $\overline{Q} \in \mathrm{NP}/\mathrm{poly}$ and hence, $\mathrm{coNP} \subseteq \mathrm{NP}/\mathrm{poly}$. This yields our claim, as then $\mathrm{PH} = \Sigma_3^{\mathrm{P}}$ by a result of Yap [11, Theorem 2]. Note that for all $\bar{x} = (x_1, \ldots, x_t)$ we have

$$\mathbb{D}(\bar{x}) \in \overline{Q'} \iff \text{for all } i \in [t] :\ x_i \in \overline{Q}. \tag{8}$$

To prove $\overline{Q} \in \mathrm{NP}/\mathrm{poly}$ it suffices to show that for sufficiently large $n \in \mathbb{N}$ there is a $t = n^{O(1)}$ and a set S of strings with $\|S\| := \sum_{x \in S} |x| = n^{O(1)}$ such that for all $x \in \{0,1\}^n$

$$x \in \overline{Q} \iff \exists x_1, \ldots, x_t \in \{0,1\}^n :\quad \Big(x \in \{x_1, \ldots, x_t\} \text{ and } \mathbb{D}(x_1, \ldots, x_t) \in S\Big).$$

In other words, S can be viewed as a polynomial size advice string for instances of length n. As we will see, the elements of S are strings in $\overline{Q'}$, more precisely, we will choose $\mathbb{D}$-values "with many preimages."

For every $m \in \mathbb{N}$, we have $|\{0,1\}^{\leq m}| \leq 2^{m+1}$, in particular,

$$|\{0,1\}^{\leq f(m) \cdot n^c}| \leq 2^{f(m) \cdot n^c + 1} \tag{9}$$

As f is pseudo-linear, by Lemma 28 there is a constant $d \in \mathbb{N}$ such that for all sufficiently large $n \in \mathbb{N}$

$$\frac{f(n^d) \cdot n^c + 1}{n^d} \leq 1. \tag{10}$$

For $n \geq 1$ we set

$$t := n^d.$$

Then (9) and (10) imply for $Y := \overline{Q'} \cap \{0,1\}^{\leq f(t) \cdot n^c}$ that

$$|Y|^{1/t} \leq 2. \tag{11}$$

Recall that $\overline{Q}_{=n} := \overline{Q} \cap \{0,1\}^n$. By (7) we can define a function $g : (\overline{Q}_{=n})^t \to Y$ by

$$g(\bar{x}) := \mathbb{D}(\bar{x}).$$

We construct the advice string S inductively. First we let $X_0 := \overline{Q}_{=n}$. Choose $y_0 \in Y$ such that

$$g^{-1}(y_0) := \big\{\bar{x} \in X_0^t \mid g(\bar{x}) = y_0\big\}$$

contains at least $|X_0|^t / |Y|$ many tuples. Let *string*$(g^{-1}(y_0))$ be the set components of tuples in $g^{-1}(y_0)$, that is,

$$\textit{string}(g^{-1}(y_0)) := \big\{x \in X_0 \mid \text{there exists some } (x_1, \ldots, x_t) \in g^{-1}(y_0) \text{ such that } x \in \{x_1, \ldots, x_t\}\big\}.$$

It follows that $g^{-1}(y_0) \subseteq \big(\textit{string}(g^{-1}(y_0))\big)^t$ and hence

$$\big|\textit{string}(g^{-1}(y_0))\big| \geq |g^{-1}(y_0)|^{1/t} \geq \left(\frac{|X_0|^t}{|Y|}\right)^{1/t} \geq \frac{|X_0|}{2},$$

the last inequality holding by (11). If $X_0 \neq \textit{string}(g^{-1}(y_0))$, then let $X_1 := X_0 \setminus \textit{string}(g^{-1}(y_0))$. Now, we view g as a function of X_1 to Y and, by the same argument as above, we choose $y_1 \in Y$ such that $|\textit{string}(g^{-1}(y_1))| \geq |X_1|/2$. We iterate this process until we reach the first $\ell \in \mathbb{N}$ with $X_\ell = \textit{string}(g^{-1}(y_\ell))$. We let

$$S := \{y_0, \ldots, y_\ell\}.$$

Then $S \subseteq Y \subseteq \overline{Q'}$ and $|S| = \ell \leq \log |X_0| \leq n$ and thus $\|S\| \leq n \cdot f(t) \cdot n^c \leq n^{d+1}$ (by (10)). Hence $\|S\|$ is polynomially bounded in n.

We show the equivalence (9). Let $x \in \{0,1\}^n$. If $x \in \overline{Q}$, by our construction of S, there is a tuple $\bar{x}$ containing x as a component such that $g(\bar{x}) = \mathbb{D}(\bar{x}) \in S$.

Conversely, assume $x \notin \overline{Q}$. Then for every $\bar{x} := (x_1, \dots, x_t)$ with $x_1, \dots, x_t \in \{0,1\}^n$ and $x \in \{x_1, \dots, x_t\}$, we have, by (8), that $\mathbb{D}(\bar{x}) \notin \overline{Q'}$ and hence $\mathbb{D}(\bar{x}) \notin S \subseteq \overline{Q'}$. □

Proof of Lemma 30: Let $c \in \mathbb{N}$ and f be pseudo-linear, say $f(t) = O(t^{1-\varepsilon})$. Assume that (Q, κ) is a parameterized problem with a linear weak OR $\mathbb{O}$ and NP-hard Q. Assume $\Sigma_3^{\text{P}} \neq \text{PH}$. For the sake of contradiction assume that $(Q, \kappa^c \times f)$ has a linear kernelization $\mathbb{K}$. By Lemma 32 it suffices to show that Q has a linear f-distillation $\mathbb{D}$.

We define $\mathbb{D}$ on finite sequences $\bar{x} = (x_1, \dots, x_t)$ by

$$\mathbb{D}(\bar{x}) := \mathbb{K}(\mathbb{O}(\bar{x})).$$

It is clear that

$$\mathbb{D}(\bar{x}) \in Q \iff \text{for some } i \in [t]: \ \ x_i \in Q.$$

Write $n := \max_{i \in [t]} |x_i|$. Then, because $\mathbb{K}$ is a linear kernelization for $(Q, \kappa^c \times f)$,

$$|\mathbb{D}(\bar{x})| = O\Big(\kappa(\mathbb{O}(\bar{x}))^c \cdot f(|\mathbb{O}(\bar{x})|)\Big) = O(n^{O(1)} \cdot |\mathbb{O}(\bar{x})|^{1-\varepsilon}) = n^{O(1)} \cdot |\mathbb{O}(\bar{x})|^{1-\varepsilon},$$

where the second equality follow from Definition 24 (2). Now, by Definition 24 (1) we know $|\mathbb{O}(\bar{x})| = t \cdot n^{O(1)}$. Hence $|\mathbb{D}(\bar{x})| = t^{1-\varepsilon} \cdot n^{O(1)}$ and therefore $\mathbb{D}$ is a linear f-distillation from Q in itself. □

7. Lower bounds for problems with a weak OR

In the previous section we have seen how to exclude *linear* kernelizations for certain reparameterizations of parameterized problems having a linear weak OR (Lemma 30). This way we proved Theorem 26.

In this section we show how to exclude *polynomial* kernelizations of these reparameterized problems. This can be done under a weaker condition than that of having a linear weak OR:

Definition 33. Let (Q, κ) be a parameterized problem. A *weak OR for* (Q, κ) is a polynomial time algorithm $\mathbb{O}$ that for every finite tuple $\bar{x} = (x_1, \dots, x_t)$ of instances of Q outputs an instance $\mathbb{O}(\bar{x})$ of Q such that

(1) $\kappa(\mathbb{O}(\bar{x})) = (\max_{i \in [t]} |x_i|)^{O(1)}$;

(2) $\mathbb{O}(\bar{x}) \in Q$ if and only if for some $i \in [t]$: $x_i \in Q$.

Hence, the notion of weak OR is obtained from that of a linear weak OR by omitting in Definition 24 condition (1), the condition on the size of the output.

The corresponding result reads as follows:

Theorem 34. *Let (Q, κ) be a parameterized problem with a weak OR and with NP-hard Q. Unless* $\text{PH} = \Sigma_3^{\text{P}}$, *there is* no *polynomial reduction from Q to itself that assigns to every instance x of Q an instance y with*

$$|y| = \kappa(x)^{O(1)} \cdot |x|^{o(1)}.$$

Recall the reparameterization $(Q, \kappa^c \times f)$ of (Q, κ) for $c \in \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$. Clearly $(Q, \kappa^c \times f)$ has a polynomial kernelization if and only if $(Q, \kappa \times f)$, the problem for $c = 1$, has one.

For the purposes of the proof of Theorem 34 we call a function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ *good* if $f(t) = t^{o(1)}$ for all $t \in \mathbb{N}$ (that is, if we can write $f(t) = t^{1/h(t)}$ for some function $h : \mathbb{N} \to \mathbb{R}_{\geq 0}$ with $\lim_{t \to \infty} h(t) = \infty$).

The statement of Theorem 34can equivalently be formulated as:

Lemma 35. *Let (Q, κ) be a parameterized problem with a weak OR and with NP-hard Q. Then for every good $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ the problem $(Q, \kappa \times f)$ has no polynomial kernelization, unless* $\text{PH} = \Sigma_3^{\text{P}}$.

Proof: Let (Q, κ) be a parameterized problem with a weak OR $\mathbb{O}$ and with NP-hard Q. Let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be good. One easily sees that there is a good increasing function $f' : \mathbb{N} \to \mathbb{R}_{\geq 0}$ of the form

$$f'(t) = 2^{\log t / \iota(\log t)} \tag{12}$$

with a nondecreasing and unbounded function $\iota : \mathbb{N} \to \mathbb{R}_{\geq 0}$ such that $f(t) \leq f'(t)$ for all (sufficiently large) t.

Assume PH $\neq \Sigma_3^{\mathrm{P}}$. For the sake of contradiction assume also that $(Q, \kappa \times f)$ has a polynomial kernelization. Of course, then $(Q, \kappa \times f')$ has a polynomial kernelization $\mathbb{K}$. By Lemma 32 it suffices to show that Q has a linear f''-distillation $\mathbb{D}$ for some pseudo-linear $f'' : \mathbb{N} \to \mathbb{R}_{\geq 0}$.

We define $\mathbb{D}$ on finite sequences $\bar{x} = (x_1, \ldots, x_t)$ by

$$\mathbb{D}(\bar{x}) := \mathbb{K}(\mathbb{O}(\bar{x})).$$

Clearly, $(\mathbb{D}(\bar{x}) \in Q$ if and only if for some $i \in [t] : \ x_i \in Q)$. Write $n := \max_{i \in [t]} |x_i|$. We will show that for some $d \in \mathbb{N}$ we have

$$|\mathbb{D}(\bar{x})| \leq (f'(t) \cdot n)^d. \tag{13}$$

Hence, $\mathbb{D}$ is a linear f''-distillation for $f''(t) := f'(t)^d$ of Q in itself.We thus get our desired contradiction, because f'' is pseudo-linear.

As $\mathbb{K}$ is a polynomial kernelization of $(Q, \kappa \times f')$, we know

$$|\mathbb{D}(\bar{x})| = \Big(\kappa(\mathbb{O}(\bar{x})) \cdot f'(|\mathbb{O}(\bar{x})|)\Big)^{O(1)} = n^{O(1)} \cdot f'(|\mathbb{O}(\bar{x})|)^{O(1)},$$

where the last equality holds by Definition 33 (1). To prove (13) it suffices to show that $f'(|\mathbb{O}(\bar{x})|) = (f'(t) \cdot n)^{O(1)}$. As $\mathbb{O}$ is polynomial time computable we know $|\mathbb{O}(\bar{x})| \leq t^c \cdot n^c$ for some constant $c \in \mathbb{N}$. Since f' is increasing, it is enough to show

$$f'(t^c \cdot n^c) \leq (f'(t) \cdot n)^{2c}.$$

By (12)

$$f'(t^c \cdot n^c) = 2^{\frac{c \cdot \log t + c \cdot \log n}{\iota(c \cdot \log t + c \cdot \log n)}}.$$

We distinguish two cases.

- If $t \geq n$, then, as ι is nondecreasing, we get

$$f'(t^c \cdot n^c) \leq 2^{\frac{2c \cdot \log t}{\iota(\log t)}} = f'(t)^{2c}.$$

- If $t < n$, then

$$f'(t^c \cdot n^c) \leq 2^{2c \cdot \log n} = n^{2c}.$$

□

Remark 36. Let $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$. One easily verifies that the property of good functions needed in the previous section (see Lemma 28) holds for all the functions $t \mapsto f^c(t)$ with $c \in \mathbb{N}$ if and only if for all $c \in \mathbb{N}$ there is a $d \in \mathbb{N}$ such that for sufficiently large n we have

$$f(n^d)^c \cdot n^c + 1 \leq n^d.$$

We show that this implies that f is "nearly" good. Let $h : \mathbb{N} \to \mathbb{R}$ be a function such that $f(t) = t^{1/h(t)}$ for every $t \in \mathbb{N}$. Then for every $c \in \mathbb{N}$ we have $h(t) > c$ for infinitely many t. Why? We know that there are $n_0, d \in \mathbb{N}$ such that for all $n \geq n_0$

$$f(n^d)^c = n^{c \cdot d / h(n^d)} < n^d.$$

Then $h(t) > c$ for $t = n_0^d, (n_0 + 1)^d, (n_0 + 2)^d, \ldots$. ⊣

References

[1] Y. Chen and J. Flum. On parameterized path and chordless path problems. In *Proceedings of the 22nd IEEE Conference on Computational Complexity (CCC'07)*, page 250 – 263, 2007

[2] Y. Chen and J. Flum. Subexponential time and fixed-parameter tractability: exploiting the miniaturization mapping. In *Proceedings of the 21st International Workshop on Computer Science Logic (CSL'07)*, Lecture Notes in Computer Science 4646, page 389 – 404, 2007.

[3] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. *On Problems Without Polynomial Kernels*, in preparation, 2007.

[4] J. Flum and M. Grohe. *Parameterized Complexity Theory*, Springer, 2006.

[5] L. Fortnow and Santhanam. Infeasibility of instance compression and succinct PCPs for NP. TR07-096 in ECCC Reports 2007, available at `http://eccc.hpi-web.de/eccc-local/Lists/TR-2007.html`

[6] M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130:3 – 31, 2004.

[7] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, Vol. 38, No. 1, 2007.

[8] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications, In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, page 719 – 728, 2006. Full version appears as TR06-022 in ECCC Reports 2006, available at `http://eccc.hpi-web.de/eccc-local/Lists/TR-2006.html`

[9] R. Impagliazzo, R.Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512 – 530, 2001.

[10] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006.

[11] C. K. Yap. Some consequences of non-uniform conditions on uniform classes, *Theoretical Computer Science* 26, page 287 – 300, 1983.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/