Lower Bounds for Kernelization

Hans L. Bodlaender(✉)

Department of Information and Computing Science,
Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
h.l.bodlaender@uu.nl

Abstract. Kernelization is the process of transforming the input of a combinatorial decision problem to an equivalent instance, with a guarantee on the size of the resulting instances as a function of a parameter. Recent techniques from the field of fixed parameter complexity and tractability allow to give lower bounds for such kernels. In particular, it is discussed how one can show for parameterized problems that these do not have polynomial kernels, under the assumption that $coNP \not\subseteq NP/poly$.

1 Introduction

In this paper, a number of recent techniques for lower bounds for kernelization are surveyed. The study of kernelization is motivated in two ways: first, it allows a precise mathematical analysis what can be achieved with polynomial time preprocessing of combinatorial problems. Second, a kernelization algorithm for a (decidable) problem also gives that the problem is fixed parameter tractable.

An important driving force behind much algorithm research is the intractability of many (combinatorial) problems, coming from practical applications and from theoretical investigations. One of the approaches when we ask for exact solutions is to first preprocess the instances before applying an exact solver: the former is typically fast, and the latter is typically slow (e.g., using integer linear programming, branch and bound, a SAT-solver). In kernelization, we make the assumption that the preprocessing takes polynomial time, is *safe* (in the sense that the answer for the problem instance is the same as, or can be derived from, the answer for the reduced instance), and we ask if there is a guaranteed upper bound of the size of the reduced instance. This upper bound is expressed as a function of some parameter of the input, possibly the target value, or some structural parameter of the input.

Parameterization is very useful for the analysis of preprocessing. The following lemma illustrates the limitations of a setting without parameterization.

Lemma 1 (Folklore). *Let Q be an NP-hard decision problem. If we have a polynomial time procedure, that given an input s, either decides if $s \in Q$, or produces an input s' with $s \in Q \Leftrightarrow s' \in Q$, then $P = NP$.*

M. Cygan and P. Heggernes (Eds.): IPEC 2014, LNCS 8894, pp. 1–14, 2014.
DOI: 10.1007/978-3-319-13524-3_1

Proof. Suppose we have such Q. Given an input, repeatedly apply Q on its own output till we decide. This gives a polynomial time algorithm for an NP-hard problem. □

The study of fixed parameter tractability is motivated from the observation that often, when we have a problem that is intractable, actual instances may be much easier due to the fact that some parameter of these instances is small. Again, this parameter may be the target value, or some structural parameter of the input. E.g., a combinatorial problem arising from facility location problem may be NP-hard, but may be still polynomial time solvable when we know that the number of facilities to be placed is at most three (e.g., by exhaustive search). Another example is that many NP-hard problems become linear time solvable on graphs of bounded treewidth (see e.g., [6].) A decidable problem has a kernel, if and only if it is fixed parameter tractable (see Lemma 2). This strong relationship between the notions of kernelization and FPT is an important motivation behind the research on kernelization. Kernels of smaller size lead to faster FPT algorithms, and thus an important question is: what is the smallest size that we can obtain for a kernel for some given parameterized problem?

For several parameterized problems, kernels of small size are known: e.g.: Vertex Cover has a kernel with at most $2k$ vertices (and $O(k^2)$ bits) (see e.g. [1,34]) Feedback Vertex Set has a kernel with $O(k^2)$ vertices (and $O(k^2)$ bits) [36]. There nowadays are many problems for which kernels of polynomial size are known. But also, for many problems, no such kernels are known. Current lower bound techniques explain why: it is shown that the problem has no polynomial kernel (or no kernel at all) unless a currently widely believed complexity theoretic assumption does not hold. Such lower bounds are useful for the algorithm designer: like an NP-hardness proofs guides us away from trying to design a polynomial time algorithm for a problem, here lower bounds can guide us away from trying to design (polynomial) kernels.

In this survey, we discuss a number of techniques to show such conditional lower bounds for kernelization.

2 Preliminaries

Throughout the paper, we assume that Σ is some finite alphabet. For a string $s \in \Sigma^*$, we denote with $|s|$ its length. A parameterized problem is a subset of $\Sigma^* \times \mathbf{N}$.

To a parameterized problem $Q \subseteq \Sigma^* \times \mathbf{N}$ we associate its *classic variant* $Q^c \subseteq (\Sigma \cup \{(,),)\})^*$, which is obtained from Q by writing the second parameter (k) in unary.

In the literature, small variations on the definition of FPT are used. We use here the notion of strongly uniform FPT (see [18, Section 2.1]).

Definition 1. *FPT (*fixed parameter tractable*) is the class of parameterized problems Q such that there is an algorithm A, that decides for a given instance*

$(s,k) \in \Sigma^* \times \mathbf{N}$ *if* $(s,k) \in Q$ *in* $O(f(k) \cdot |s|^c)$ *time, for a computable function* f *and constant* $c \in \mathbf{N}$.

Many parameterized problems are known to be fixed parameter tractable (in FPT); the design of efficient parameterized algorithms is a very active field of study. E.g., see [18,19,21,35].

Definition 2. *A* kernelization algorithm *or in short, a* kernel *for a parameterized problem* Q *is an algorithm* A*, that, given an instance* $(s,k) \in \Sigma^* \times \mathbf{N}$*, outputs an instance* $(s',k') \in \Sigma^* \times \mathbf{N}$*, such that there are computable functions* f *and* g*, and a constant* c*, with*

1. A *uses* $O(f(k)|s|^c)$ *time;*
2. $(s,k) \in A$*, if and only if* $(s',k') \in A$*;*
3. $|s'| \leq g(k)$*,* $k' \leq g(k)$*.*

Thus, a kernelization algorithm is a polynomial time algorithm, that transforms an input for parameterized problem Q to an equivalent input, but with the size of the latter bounded by a (computable) function in the parameter. The function g is said to be the *size* of the kernel.

There are minor variations on the definition of kernelization, and also more general notions have been studied. Some of these are reviewed in Sect. 6. The notion of kernelization is tightly bound to the notion of fixed parameter tractability, as the following well known result shows.

Lemma 2 (Folklore). *Let* $Q \subseteq \Sigma^* \times \mathbf{N}$ *be a decidable problem. Then* $Q \in FPT$*, if and only if* Q *has a kernel.*

Proof. Suppose $Q \in FPT$. Let A be an algorithm that decides on Q in $f(k)n^c$ time, for some computable function f and constant c. Suppose we have an instance (x,k) of size n. Run algorithm A for n^{c+1} steps. If A terminates within this time, then output a trivial $O(1)$ size yes- or no-instance. Otherwise, $n \geq f(k)$: output (x,k). This fulfils the definition of a kernel.

Suppose we have a kernelization algorithm A. First run A on the input, and then, as Q is decidable, run any decision algorithm on the remaining reduced instance. The time of the latter step is bounded by a function of the parameter k, and the time of the former step is polynomial. The combination of the steps is an FPT algorithm. □

Lemma 2 is important for two reasons. First, it shows us that we can turn a kernelization algorithm directly in an FPT algorithm, and the method (first build a small equivalent instance, and then solve that instance) follows the approach discussed in the introduction for hard problems. Kernels of smaller size give faster FPT algorithms, and this thus motivates the search for kernels of small size. Second, it shows that if we have reason to believe that no FPT algorithm exists for a problem Q, then we also have reason to believe that there is no kernelization algorithm for Q. The latter is precisely the case for parameterized problems that are $W[1]$-hard. If a $W[1]$-hard problem belongs to the class FPT, then we

have that $FPT = W[1]$, and from that it follows that the Exponential Time Hypothesis (ETH) does not hold [10]. Thus a corollary of Lemma 2 is that no $W[1]$-hard problem has a kernel, assuming the ETH.

As discussed above, we are interested in kernels of small size. An important class of kernels are the *polynomial kernels*: a kernel is polynomial, if the function g in Definition 2 (i.e., the upper bound for the resulting instances (s', k') on $|s'|$ and of k') is polynomial, i.e., there is a constant c' with $g(n) = O(n^{c'})$.

In this survey, we look at a number of techniques to give conditional proofs that such polynomial kernels do not exist. The results usually depend on the assumption that $NP \not\subseteq coNP/poly$, or, equivalently, that $coNP \not\subseteq NP/poly$. If this assumption would not hold, the polynomial time hierarchy would collapse to its third level [37]. For many parameterized problem, unconditional proofs that no polynomial kernel exists cannot reasonably be expected. E.g., if $P = NP$, then the parameterized variants of NP-complete problems (like Long Path, Treewidth) have kernels of size $O(1)$.

3 Compositions

In this section, we discuss the first technique to show that problems do not have polynomial kernels: *composition*. Actually, composition comes in two flavours: OR-composition, and AND-composition. In several cases, compositionality gives simple and sometimes even trivial proofs for parameterized problems that they do not have polynomial kernels, assuming $NP \not\subseteq coNP/poly$. In several other cases, such proofs can be hard and lengthy. We start this section with describing the intuition behind the ideas. Then we introduce the main notions, stating the main theorems, and proving the main theorem for the case of OR-composition. We end the section with showing for some parameterized problems that they are compositional, and conclude that they have no polynomial kernel, again under the assumption that $NP \not\subseteq coNP/poly$.

3.1 Intuition

To get the intuition behind the approach, we consider the Long Path problem. A k-path is a simple path with at least k edges. In the Long Path problem, we are given an undirected graph G, integer k, and must decide if G has a k-path; k is the parameter.

Let us look at the situation that we would have a kernel of polynomial size for the Long Path problem, say a polynomial time algorithm A that reduces an instance of Long Path to one with k^c bits to describe it. Now, suppose we have a graph with k^{c+1} connected components. G contains a simple path with k edges, if and only if at least one of its connected components has a simple path with k edges. But these different connected components can be regarded as separate instances of Long Path, and thus, we would have a manner of reducing k^{c+1} different instances of Long Path to k^c bits in total: significantly fewer bits

than the number of components. Such reduction seems only possible when we are able to *solve* the Long Path problem on some of the components; that is unlikely in polynomial time as the problem in its classic variant is NP-complete.

With this intuition in mind, we now look at the formal notion of compositions, and see how this can be used to prove conditional lower bounds for kernelization in Sect. 3.2.

3.2 Compositionality and Lower Bounds

The techniques and results in this section are mostly due to Bodlaender et al. [3] and Fortnow and Santhanam [22].

The notion of composition comes in two flavours: or-composition and and-composition. We give the definition of or-composition in full, and explain the difference with the definition of and-composition.

Definition 3. *An* or-composition *for a parameterized problem* $Q \subseteq \Sigma^* \times \mathbf{N}$ *is an algorithm* A *that gets as input a sequence of instances* $(s_1, k), \ldots (s_r, k)$*, and outputs one instance* (s', k')*, such that*

1. *A uses time that is bounded by a polynomial in* $\sum_{i=1}^{r} |s_i| + k$*;*
2. $(s', k') \in Q$*, if and only if there exists an* i*,* $1 \leq i \leq k$ *with* $(s_i, k) \in A$*;*
3. k' *is bounded by a polynomial in* k*.*

I.e., the or-composition algorithm transforms a sequence of instances with the same parameter to one instance, the latter being a yes-instance for Q if and only if at least one of the former instances is a yes-instance; it uses time that is polynomial in the total size of the instances in the sequence; and the resulting parameter must be polynomially bounded in the parameter in the original instances. We see examples of compositions in Sect. 3.3.

And-compositions are defined in the same way; we change the second condition in Definition 3 to

– $(s', k') \in Q$, if and only if for all i, $1 \leq i \leq k$ with $(s_i, k) \in A$.

If a problem has an or-composition, we say it is *or-compositional*; similarly for *and-compositional*. Combining the results of three different papers, we obtain the following central result, which provides us with a powerful tool to show that problems are likely not to have a polynomial kernel. In Sect. 3.4 we sketch the proof for part (a).

Theorem 1. *(a)* [Bodlaender et al. [3], Fortnow and Santhanam [22]] *Let Q be a parameterized problem that is or-compositional and whose classic variant is NP-hard. Then Q does not have a polynomial kernel unless $coNP \subseteq NP/poly$. (b)* [Bodlaender et al. [3], Drucker [20]] *Let Q be a parameterized problem that is and-compositional and whose classic variant is NP-hard. Then Q does not have a polynomial kernel unless $coNP \subseteq NP/poly$.*

3.3 Compositional Problems

As a graph has a k-path, if and only if at least one of its connected components has a k-path, the Long Path problem is trivially or-compositional: just take the disjoint union of the graphs of the instances, and do not change parameter k. Similarly, if we have a graph parameter like Treewidth which is for each graph the maximum of the parameters value over all connected components, disjoint union gives a trivial and-composition. A direct corollary of Theorem 1 and the corresponding NP-hardness results is that Long Path and Treewidth have no polynomial kernel unless $coNP \subseteq NP/poly$.

In many other cases, compositions are far from trivial. See for example [13,17,28,32].

Compositionality of Disjoint Factors. An example of a non-trivial or-composition is the following, from Bodlaender et al. [8]. In the Disjoint Factors problem, we are given a string $s \in \{1, \ldots, k\}$, and ask for a collection of k substrings $s_1, \ldots, s_k$ of s, that do not overlap, and for each i, s_i starts and ends with an i. The size of the alphabet k is the parameter of this problem.

For example, 1231331212 is a positive instance, with substrings 1231, 212 and 33; and 1221 is a negative instance.

An or-composition for Disjoint Factors can be obtained as follows. First, we notice that the problem is solvable in $O(kn \cdot 2^k)$ time with standard dynamic programming techniques for strings of length n. Suppose we have instances $s^1, \ldots, s^r \in \{1, 2, \ldots, k\}$. If $r > 2^k$, we can solve all these instances in polynomial time, so assume $r \leq 2^k$. By possibly adding dummy instances, we can assume that $r = 2^k$. Now, add $k+1, \ldots, 2k$ to our alphabet. Instead of formally defining the composition, the following examples will make the scheme hopefully clear: if $k = 2$, take $34s_1 4s_2 43s_4 3$; if $k = 3$, take

$$456s + 16s_2 656s_3 6s_4 65456s_5 6s_6 656s_7 6s_8 654.$$

The resulting string has a solution, if and only if at least one s_i has a solution: the factor with $k+1$ 'disables' either the first or second half of the strings s_i; the next factor disables half of the remaining ones, and once we found factors for $k+1$ till $2k$, only one s_i remains to find the factors for $1, \ldots, k$. For a more detailed explanation, see [8] or [33]. Now, as Disjoint Factors is NP-complete [8], it follows from Theorem 1, that it has no polynomial kernel unless $coNP \subseteq NP/poly$.

3.4 Proof Sketch For Lower Bounds With Or-Composition

Below, we give a proof for Theorem 1(a). The original proof was via a notion of distillation; we give here a direct proof without this intermediate step; it follows the proof method from [22, Theorem3.1]. Druckers proof [20] for the case of and-composition is much more involved; a new and possibly simpler proof was very recently given by Dell [14].

Proof of Theorem 1(a). Suppose we have a parameterized problem Q that is compositional, whose classic variant is NP-hard, and that has a kernel with inputs with parameter k mapped to an equivalent input with at most k^c bits for some constant c.

Throughout the following proof, we will switch without notification between the parameterized and classic variants of the problem, and ignore some simple but technical details on how instances of the two versions are mapped. We always assume that the parameter is given in unary.

Denote the complement of the classic variant of Q by not-Q. (E.g., if Q is the problem, given a graph G and integer parameter k, to decide if G has a simple cycle of length at least k, then not-Q is the problem, given a pair (G, k) to decide if all simple cycles in G have length at most $k-1$.) As (the classic variant of) Q is NP-hard, we have that not-Q is coNP-hard. Thus, if we show that not-Q belongs to $NP/poly$, we have that $coNP \subseteq NP/poly$ and proved the result.

Hence, what remains to be done is to give a nondeterministic algorithm for not-Q that can access polynomial advice. I.e., for each input size n, the algorithm can consult a string advice(n), whose size is bounded by a polynomial in n. In our case, the advice will consist of $O(n^2)$ instances, each of size at most $n^{cc'}$, thus the advice has size $O(n^{cc'+2})$ bits. Each element of the advice will belong to not-Q.

The algorithm will have the following form:

- Suppose an instance (x, k) of size n is given. (We have that $k \leq n$.)
- Set $r = n^{cc'}$.
- Non-deterministically guess a sequence of r instances, each of size n, and with parameter k.
- If (x, k) is not in the sequence, then reject.
- Compute the composition of the sequence, say (y, k'). By assumption on the composition, we have that $k' \leq k^{c'} \leq n^{c'}$.
- Compute the kernel (z, k'') of (y, k'). By assumption on the kernelization algorithm, we have that the size of this instance is at most $k'^c \leq n^{cc'} = r$.
- Check if (z, k'') is in advice(n). If so, accept; otherwise, reject.

Each element in the advice will be a negative instance of Q (or, equivalently, a positive instance of not-Q.) If $(x, k) \in Q$, then the properties of or-composition and kernelization imply that $(y, k') \in Q$ and thus that $(z, k'') \in Q$. As the advice only contains elements from not-Q, $(z, k'') \notin$ advice(n), and we correctly reject. What remains now is to show that there is a sufficiently small advice set such that for each $(x, k) \in$ not-Q of size n, there is a guess that gives a kernel in the advice. Lemma 3 shows that such a set indeed exists.

We say that an instance $(x, k) \in$ not-Q of size n is *covered* by an instance (y, k), if there is a sequence $\mathbf{x} = (x_1, k), (x_2, k), \ldots, (x_{k^c}, k)$ such that

1. There is an i, $1 \leq i \leq k^c$ with $x = x_i$. (I.e., (x, k) is part of the sequence.
2. The kernelization algorithm, applied to the result of the or-composition algorithm, applied to the sequence $\mathbf{x}$ belongs to advice(n).

Lemma 3. *For sufficiently large n, there exists a set advice(n) of $O(n^2)$ instances of size at most $n^{cc'}$, such that*

- *Each instance $(y, k') \in$ advice(n) belongs to not-Q.*
- *Each instance (x, k) of size n is covered by an element of advice(n).*

Proof. We build the advice incrementally, starting with an empty set. Repeat the following step: add to the advice an instance of size at most $n^{cc'}$ in not-Q that covers the largest number of instances from not-Q of size n that are not yet covered by the advice.

Claim. There is an instance in not-Q of size at most $n^{cc'}$ that covers a constant fraction of all uncovered instances of size n in not-S.

Proof. Recall that $r = n^{cc'}$. Let A be the set of uncovered instances of size n in not-S. These form $|A|^r$ r-tuples. Each tuple is mapped by composition and kernelization to an instance of size at most r, of which there less than $2 \cdot 2^r$. By pigeon-hole principle, one of the latter is the image of $\frac{|A|^r}{2 \cdot 2^r}$ tuples, and thus covers at least $(\frac{|A|^r}{2 \cdot 2^r})^{1/r} \geq |A|/4$ instances from A. □

As there are at most 2^n instances of size n in not-S, the claim shows that $O(\log 2^n) = O(n)$ elements are sufficient for the advice. □

Lemma 3 shows that the advice is polynomial, and thus completes the proof of Theorem 1(a).

4 Transformations

A second technique to show conditional lower bounds for kernels is based upon using transformations. The technique is quite similar to usual NP-completeness proofs, with the specific twist here that the transformation should map an instance with parameter k to a new instance whose parameter is polynomially bounded in k. The technique was independently observed by several groups of authors [2,8,17]; the formalization is taken from [8], while the terminology was proposed by Lokshtanov.

Definition 4. *A* polynomial parameter transformation (ppt) *from a parameterized problem Q to a parameterized problem R is an algorithm A, that given an instance of Q, outputs an instance of R, such that*

1. *For all instances (x, k), $(x, k) \in Q \Leftrightarrow A(x, k) \in R$.*
2. *Given an instance (x, k), A uses time, polynomial in $|x| + k$.*
3. *There is a constant c, such that for all instances (x, k), if $A(x, k) = (x', k')$, then $k' \leq k^c$.*

The following theorem can be easily proven, and gives a direct method to lift lower bounds for kernels for some problem to similar lower bounds for other problems. See e.g. [8] or [5] for the proof.

Theorem 2. *If there is a ppt from Q to R and a polynomial time reduction from the classic variant of R to the classic variant of Q, and R has a polynomial kernel, then Q has a polynomial kernel.*

Corollary 1. *Suppose we have parameterized problems Q and R, with the classic variant of Q NP-hard, and the classic variant of R in NP. If Q has no polynomial kernel, then R has no polynomial kernel.*

As an example, we consider the Disjoint Cycles problem: determine, given an undirected graph G and integer k (the parameter), whether G has at least k vertex disjoint cycles. This problem is well known to be NP-complete. A ppt from Disjoint Cycles to Disjoint Factors is as follows. Given a string $s_1 s_2 \cdots s_n \in \{1, 2, \ldots, k\}^n$, we build a graph with $n + k$ vertices. We first take a path with n vertices, each vertex representing a character from the string. For each symbol in the alphabet $i \in \{1, \ldots, k\}$, we add a vertex v_i, and make v_i incident to all path vertices that represent a character with this symbol. See Fig. 1. It is not hard to observe that the resulting graph has k disjoint cycles, if and only if the string has the required set of factors. (Each cycle needs to use one of the vertices not on the path, the remainder of the cycle corresponds to a factor, and as the cycles must be disjoint, the factors may not overlap.) From the earlier observed lower bound for kernels for Disjoint Factors, it thus follows that Disjoint Cycles has no polynomial kernel assuming $coNP \not\subseteq NP/poly$,

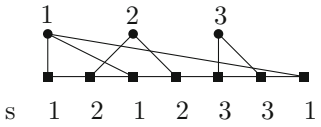

Fig. 1. Example of transformation: Disjoint Factors to Disjoint Cycles

5 Cross Composition

Suppose we want to show that parameterized problem Q has no polynomial kernel under the usual assumption that $NP \not\subseteq coNP/poly$. An (and- or or-) composition as discussed in the previous chapter takes a collection of instances of Q and 'merges' these to one instance of the problem. The notion of cross composition, introduced by Bodlaender et al. [5], allows to start with a collection of instances of some (other) problem Q' (which should be NP-hard), and transforms this collection to one instance of Q. In this way, the notion of cross composition gives a more powerful tool to proof the conditional lower bounds for kernels.

The first ingredient for the notion of cross composition is a polynomial equivalence relation.

Definition 5. *A* polynomial equivalence relation *is an equivalence relation $\sim$ on Σ^*, such that*

- *Given two strings s_1 and s_2, we can decide if $s_1 \sim s_2$ in time, polynomial in $|s_1| + |s_2|$.*
- *There is a polynomial p, such that The number of equivalence classes of $\sim$ that contain strings of length at most r is bounded by $p(r)$.*

An example is the following. We consider instances of a graph problem, and two instances are equivalent if and only if they have the same number of vertices and the same number of edges.

We now come to the definition of OR cross composition, and then briefly give the difference with the definition of AND cross composition.

Definition 6. *Suppose we have a parameterized problem $Q \subset \Sigma^* \mathbf{N}$, a language $L \subseteq \Sigma^*$ and a polynomial equivalence relation $\sim$ on Σ^*. An* OR cross composition *of L to Q with respect to $\sim$ is an algorithm A, such that*

- *The input of A is a sequence instances $s_1, \ldots, s_r$ of L that belong to the same equivalence class of $\sim$.*
- *A uses time, polynomial in $\sum_{i=1}^{r} |s_i|$.*
- *A outputs one instance (s', k') of Q, with k' bounded by a polynomial in $\max |s_i| + \log k$.*
- *$(s', k') \in Q$ if and only if there is an i with (s_i) in L.*

The notion of *AND cross composition* is defined in exactly the same way, except that the last condition in Definition 6 is replaced as follows.

$(s', k') \in Q$ if and only if for all an i with (s_i) in L.

Building upon the techniques and results of Bodlaender et al. [3], Fortnow and Santhanam [22] and Drucker [20], Bodlaender, Jansen, and Kratsch [5] obtained the following result, which provides us with a powerful mechanism to show conditional kernel lower bounds.

Theorem 3 (Bodlaender et al. [5]). *Let $L \subseteq \Sigma^*$ be an NP-hard language, let $\sim$ be a polynomial equivalence relation, and $Q \subseteq \Sigma^* \times \mathbf{N}$ be a parameterized language. Suppose there exists an OR cross composition or an AND cross composition from L to Q with respect to $\sim$. Then Q does not have a polynomial kernel, unless $NP \subseteq coNP/poly$.*

For three reasons, this result gives more possibilities to show conditional kernel lower bounds:

- We can start with a collection of instances of any NP-hard problem, instead of having to use instances of the problem we want to prove a bound for itself.
- The polynomial equivalence relation allows us to make several additional assumptions on this collection of instances.
- The bound on k' helps to bound the number of instances of L we have to compose.

Cross compositions were used for kernel lower bounds for e.g., Treewidth and Pathwidth [4], Clique Cover [12], Vertex Cover [28], and Test Cover [23].

6 Other Models and Extensions

This survey so far only discussed results that some problems do not have polynomial kernels under the assumption of $coNP \not\subseteq NP/poly$. Several extensions and variants have been studied, that will be briefly mentioned below.

Many if not all of the lower bounds we discussed hold for more general settings: when we compress instances of a compositional language Q into instances of some other language R, compression in a setting of protocols, etc.

Many of the lower bounds also hold when we compress to a different target language. Also, lower bounds can be stated for the amount of information sent in certain types of protocols, for details see e.g. [16]. Drucker also proved his lower bounds for more general settings, e.g., probabilistic and quantum [20].

Earlier, lower bounds for a different model of compression with applications in e.g., cryptography were investigated by Harnik and Naor [24].

Non-increasing parameters. In interesting technique for lower bounds for kernels that do not increase the parameter was introduced by Chen et al. [11]. Combining composition and branching, one can show for several problems (including Long Path) that they do not have a polynomial kernel which does not increase the parameter, assuming that $P \neq NP$. Chen et al. [9] also augment the framework discussed in Sect. 3.2 to obtain stronger lower bounds.

Co-nondeterminism. Kratsch [29] and Kratsch et al. [31] showed that one can use *co-nondeterministic composition* to prove lower bounds, thus extending the power of the framework to a more general notion of composition. E.g., [29] gives a lower bound for kernels for the problem to decide whether a given graph contains a vertex set of size k that is independent or a clique; the problem is in FPT as direct consequence of Ramsey theory.

6.1 Polynomial Lower Bounds

An important result was obtained by Dell and van Melkebeek [16], who extended the techniques to obtain sharp polynomial lower bounds for problems with a polynomial kernel for several well known parameterized problems.

For instance, they showed that if there is a polynomial time algorithm that gives a kernel for Vertex Cover or Feedback Vertex Set with $O(k^{2-\epsilon})$ bits for some $\epsilon > 0$, then $coNP \subseteq NP/poly$. Thus we have existing kernels for these problems (see e.g. [1,36]) are to be expected to be sharp with respect to the number of edges. The technique has been used by Dell and Marx [15] to obtain lower bounds for kernels for a number of packing problems. See also [26].

An interesting application of the technique was found by Kratsch et al. [30], who show that a simple quadratic kernel for the problem to cover a point set in the plane with k straight lines is essentially tight.

Parametric duality. Chen et al. [9] introduce the technique of parametric duality, which allows to give linear lower bounds for several problems, e.g., a bound of $2k$ vertices for Vertex Cover, assuming that $P \neq NP$.

6.2 Turing Kernelization

A different model for kernelization is Turing kernelization. A Turing kernel is an algorithm that solves a parameterized problem in polynomial time, but the algorithm in addition has access to an oracle that decides instances of size $f(k)$ in one time step. With a proof similar to that of Lemma 2, one sees that a decidable problem has a Turing kernel iff it is in FPT.

There are several problems that have a polynomial Turing kernel, but (assuming $coNP \not\subseteq NP/poly$) polynomial kernel. See e.g., [2,27]. A lower bound theory for Turing kernelization has been set up by Hermelin et al. [25].

7 Conclusions

This paper gives a compact and incomplete survey on a number of techniques to show lower bounds (under a complexity theoretic assumption) for kernels were discussed. An excellent and more extensive survey was made by Misra et al. [33].

The framework helps in the classification of the complexity of parameterized problems: is the problem in P (regardless of parameter), has it a polynomial kernel, does it belong to FPT, is it in XP, or is it already NP-hard for some fixed value of the parameter?

I would like to end the survey with a practical warning: even when we know that for parameter, the problem at hand has no polynomial kernel, it still can be very useful to preprocess the problem. An illustrative example is for the problem of Treewidth. Treewidth parameterized by the target value is and-compositional, and thus not likely to have a polynomial kernel. However, preprocessing rules were seen to be very effective for many instances from real-world applications [7]. In [4,26], (part of) a theoretical explanation is provided: with different parameterizations, treewidth has kernels of small (polynomial) size.

Acknowledgments. This survey would not have been possible without the discussions and cooperation with several collegues: Rod Downey, Mike Fellows, Bart Jansen, Danny Hermelin, Stefan Kratsch, Stéphan Thomassé and Andres Yeo. Thank you very much! I apologize to all whose work was inadvertingly or due to space was not or insufficiently discussed here.

References

1. Abu-Khzam, F.N., Collins, R.L., Fellows, M.R., Langston, M.A., Suters, W.H., Symons, C.T.: Kernelization algorithms for the vertex cover problem: theory and experiments. In: Proceedings of the 6th Workshop on Algorithm Engineering and Experimentation and the 1st Workshop on Analytic Algorithmics and Combinatorics, ALENEX/ANALCO 2004, pp. 62–69. ACM-SIAM (2004)
2. Binkele-Raible, D., Fernau, H., Fomin, F.V., Lokshtanov, D., Saurabh, S., Villanger, Y.: Kernel(s) for problems with no kernel: on out-trees with many leaves. ACM Trans. Algorithms **8**(5), 38 (2012)
3. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. J. Comput. Syst. Sci. **75**, 423–434 (2009)

4. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Preprocessing for treewidth: a combinatorial analysis through kernelization. SIAM J. Discrete Math. **27**, 2108–2142 (2013)
5. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. SIAM J. Discrete Math. **28**, 277–305 (2014)
6. Bodlaender, H.L., Koster, A.M.C.A.: Combinatorial optimization on graphs of bounded treewidth. Comput. J. **51**(3), 255–269 (2008)
7. Bodlaender, H.L., Koster, A.M.C.A., Eijkhof, F.: Pre-processing rules for triangulation of probabilistic networks. Comput. Intell. **21**(3), 286–305 (2005)
8. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. Theoret. Comput. Sci. **412**, 4570–4578 (2011)
9. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: lower bounds and upper bounds on kernel size. SIAM J. Comput. **37**, 1077–1106 (2007)
10. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Strong computational lower bounds via parameterized complexity. J. Comput. Syst. Sci. **72**, 1346–1367 (2006)
11. Chen, Y., Flum, J., Müller, M.: Lower bounds for kernelizations and other preprocessing procedures. Theory Comput. Syst. **48**(4), 803–839 (2011)
12. Cygan, M., Kratsch, S., Pilipczuk, M., Pilipczuk, M., Wahlström, M.: Clique cover and graph separation: new incompressibility results. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 254–265. Springer, Heidelberg (2012)
13. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Kernelization hardness of connectivity problems in d-degenerate graphs. Discrete Appl. Math. **160**, 2131–2141 (2012)
14. Dell, H.: A simple proof that AND-compression of NP-complete problems is hard. In: Proceedings IPEC 2014 (2014)
15. Dell, H., Marx, D.: Kernelization of packing problems. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, pp. 68–81 (2012)
16. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In: Schulman, L.J. (ed.) Proceedings of the 42nd Annual Symposium on Theory of Computing, STOC 2010, pp. 251–260 (2010)
17. Dom, M., Lokshtanov, D., Saurabh, S.: Incompressibility through colors and IDs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 378–389. Springer, Heidelberg (2009)
18. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Berlin (1999)
19. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, Berlin (2013)
20. Drucker, A.: New limits to classical and quantum instance compression. In: Proceedings of the 53rd Annual Symposium on Foundations of Computer Science, FOCS 2012, pp. 609–618 (2012)
21. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
22. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. J. Comput. Syst. Sci. **77**, 91–106 (2011)
23. Gutin, G., Muciaccia, G., Yeo, A.: Non-existence of polynomial kernels for the test cover problem. Inf. Process. Lett. **113**, 123–126 (2013)
24. Harnik, D., Naor, M.: On the compressibility of $\mathcal{NP}$ instances and cryptographic applications. SIAM J. Comput. **39**, 1667–1713 (2010)

25. Hermelin, D., Kratsch, S., Sołtys, K., Wahlström, M., Wu, X.: A completeness theory for polynomial (Turing) kernelization. In: Gutin, G., Szeider, S. (eds.) IPEC 2013. LNCS, vol. 8246, pp. 202–215. Springer, Heidelberg (2013)
26. Jansen, B.M.P.: On sparsification for computing treewidth. In: Gutin, G., Szeider, S. (eds.) IPEC 2013. LNCS, vol. 8246, pp. 216–229. Springer, Heidelberg (2013)
27. Jansen, B.M.P.: Turing kernelization for finding long paths and cycles in restricted graph classes. In: Schulz, A.S., Wagner, D. (eds.) ESA 2014. LNCS, vol. 8737, pp. 579–591. Springer, Heidelberg (2014)
28. Jansen, B.M.P., Bodlaender, H.L.: Vertex cover kernelization revisited - upper and lower bounds for a refined parameter. Theory Comput. Syst. **53**, 263–299 (2013)
29. Kratsch, S.: Co-nondeterminism in compositions: a kernelization lower bound for a Ramsey-type problem. ACM Trans. Algorithms **10**(4), 19 (2014)
30. Kratsch, S., Philip, G., Ray, S.: Point line cover: the easy kernel is essentially tight. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, pp. 1596–1606 (2014)
31. Kratsch, S., Pilipczuk, M., Rai, A., Raman, V.: Kernel lower bounds using co-nondeterminism: finding induced hereditary subgraphs. In: Fomin, F.V., Kaski, P. (eds.) SWAT 2012. LNCS, vol. 7357, pp. 364–375. Springer, Heidelberg (2012)
32. Kratsch, S., Wahlström, M.: Two edge modification problems without polynomial kernels. Discrete Optim. **10**, 193–199 (2013)
33. Misra, N., Raman, V., Saurabh, S.: Lower bounds on kernelization. Discrete Optim. **8**, 110–128 (2011)
34. Nemhauser, G.L., Trotter, L.E.: Vertex packing: structural properties and algorithms. Math. Program. **8**, 232–248 (1975)
35. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, Oxford (2006)
36. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM Trans. Algorithms **6**(2), 1–8 (2010)
37. Yap, H.P.: Some Topics in Graph Theory. London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge (1986)