Kernel: Lower and Upper Bounds

Daniel Lokshtanov[*] Saket Saurabh[*]

May 21, 2009

Abstract

Through this lecture note we try to provide a portal into the emerging filed of *kernelization.* We exhibit through examples various tools to prove both lower and upper bounds on the kernel sizes.

1 Introduction

Preprocessing (data reduction or kernelization) as a strategy of coping with hard problems is universally used in almost every implementation. The history of preprocessing, like applying reduction rules to simplify truth functions, can be traced back to the 1950's [16]. A natural question in this regard is how to measure the quality of preprocessing rules proposed for a specific problem. For a long time the mathematical analysis of polynomial time preprocessing algorithms was neglected. The basic reason for this anomaly was that if we start with an instance I of an NP-hard problem and can show that in polynomial time we can replace this with an equivalent instance I' with $|I'| < |I|$ then that would imply P=NP in classical complexity. The situation changed drastically with advent of parameterized complexity. Combining tools from parameterized complexity and classical complexity it has become possible to derive upper and lower bounds on the sizes of reduced instances, or so called *kernels.* Importance of preprocessing and the mathematical challenges it poses is beautifully expressed in the following quote by Fellows.

It has become clear, however, that far from being trivial and uninteresting, that pre-processing has unexpected practical power for real world input distributions, and is mathematically a much deeper subject than has generally been understood.

Working View Point on Kernel: In parameterized complexity each problem instance comes with a parameter k and the parameterized problem is said to admit a *polynomial kernel* if there is a polynomial time algorithm (the degree of polynomial is independent of k), called a *kernelization* algorithm, that reduces the input instance down to an instance with size bounded by a polynomial $p(k)$ in k, while preserving the answer. This reduced instance is called a $p(k)$ *kernel* for the problem. If $p(k) = O(k)$, then we call it a *linear kernel.* Kernelization has been extensively studied in the realm of parameterized complexity,

[*]Department of Infomatics, University of Bergen, Norway. {daniello|saket.saurabh}@ii.uib.no.

resulting in polynomial kernels for a variety of problems. Notable examples include a $2k$-sized vertex kernel for VERTEX COVER [6], a $355k$ kernel for DOMINATING SET on planar graphs [1], which later was improved to a $67k$ kernel [5], and an $O(k^2)$ kernel for FEEDBACK VERTEX SET [18] parameterized by the solution size.

2 Basic Definitions

A parameterized problem L is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. An instance of a parameterized problem consists of (x, k), where k is called the parameter. A central notion in parameterized complexity is *fixed parameter tractability (FPT)* which means for a given instance (x, k) solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size. The notions of *kernelization* and *composition* are formally defined as follows.

Definition 1. *A* kernelization algorithm*, or in short, a kernel for a parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in time polynomial in $|x| + k$ a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that (a) $(x, k) \in Q$ if and only if $(x', k') \in Q$ and (b) $|x'| + k' \leq g(k)$, where g is an arbitrary computable function. The function g is referred to as the size of the kernel. If g is a polynomial function then we say that Q admits a polynomial kernel.*

We close with some definitions from graph theory. Let $G = (V, E)$ be a graph. For a vertex v in G, we write $N_G(v)$ to denote the set of v's neighbors in G, and we write $\deg_G(v)$ to denote the *degree* of v, that is, the number of v's neighbors in G. If it is clear from the context which graph is meant, we write $N(v)$ and $\deg(v)$, respectively, for short. A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. The subgraph G' is called an *induced subgraph* of G if $E' = \{\{u, v\} \in E \mid u, v \in V'\}$, in this case, G' is also called the subgraph *induced by* V' and denoted with $G[V']$. A vertex v *dominates* a vertex u if $u \in N(v)$.

3 Upper Bound Machinery

We illustrate the method of kernelization using the parameterized version of MAX3SAT where given a boolean 3-CNF formula and an integer parameter k, we would like to know whether there is an assignment to the variables that satisfies at least k of the clauses. Our other examples in this Section include a kernel for d-HITTING SET using the *Sunflower Lemma*, a $4k$ sized kernel for VERTEX COVER using *crown decomposition* and a 2^k kernel for (EDGE) CLIQUE COVER.

3.1 Max3Sat

Let F be a given boolean CNF 3-SAT formula with n variables and m clauses. t is well known that in any boolean CNF formula, there is an assignment that satisfies at least half of the clauses (given any assignment that doesn't satisfy half the clauses, its bitwise complement will). So if the parameter k is less than $m/2$, then there is an assignment to the variables that satisfies at least k of the clauses. Otherwise, $m \leq 2k$, and so $n \leq 6k$.

3.2 d-Hitting Set

In this Section we give a kernelization algorithm for the d-Hitting Set problem which is defined as follows:

> d-Hitting Set (d-HS) : Given a collection $\mathcal{C}$ of d element subsets of an universe U and a positive integer k, the problem is to determine whether there exists a subset $U' \subseteq U$ of size at most k such that U' contains at least one element from each set in $\mathcal{C}$.

Our kernelization algorithm is based on the following widely used *Sunflower Lemma.* We first define the terminology used in the statement of the lemma. A *sunflower* with *k petals* and a *core* Y is a collection of sets $S_1, S_2 \cdots S_k$ such that $S_i \cap S_j = Y$ for all $i \neq j$; the sets $S_i - Y$ are petals and we require that none of them be empty. Note that a family of pairwise disjoint sets is a sunflower (with an empty core).

Lemma 1 ([10])**. [Sunflower Lemma]** *Let $\mathscr{F}$ be a family of sets over an universe $\mathscr{U}$ each of cardinality s. If $|\mathscr{F}| > s!(k-1)^s$ then $\mathscr{F}$ contains a sunflower with k petals and such a sunflower can be computed in time polynomial in the size of $\mathscr{F}$ and $\mathscr{U}$.*

Now we are ready to prove the following theorem about kernelization for d-HS.

Theorem 1. *d-HS has a kernel of size $O(k^d d! d^2)$. That is, given an instance $(U, \mathcal{C}, k)$ of d-HS, we can replace it with an equivalent instance $(U, \mathcal{C}', k')$ with $|\mathcal{C}'| \leq O(k^d d! d)$ in polynomial time.*

Proof. The crucial observation is that if $\mathcal{C}$ contains a sunflower $S = \{S_1, \cdots, S_{k+1}\}$ of cardinality $k+1$ then every hitting set of $\mathcal{C}$ of size at most k must intersect with the core Y of the sunflower S, otherwise we will need hitting set of size more than k. Therefore if we let $\mathcal{C}' = (\mathcal{C} \cup Y) \setminus S$ then the instance $(U, \mathcal{C}, k)$ and $(U, \mathcal{C}', k)$ are equivalent.

Now we apply the Sunflower Lemma for all $d' \in \{1, \cdots, d\}$, repeatedly replacing sunflowers of size at least $k+1$ with their cores until the number of sets for any fixed $d' \in \{1, \cdots, d\}$ is at most $O(k^{d'} d'!)$. Summing over all d we obtain the desired kernel of size $O(k^d d! d)$. □

3.3 Crown Decomposition : Vertex Cover

In this Section we introduce a *crown decomposition* based kernelization for Vertex Cover. It is based on a connection between matchings and vertex cover which is that the maximum size of a matching is a lower bound for the minimum cardinality vertex cover. We first define Vertex Cover precisely as follows.

> Vertex Cover (VC): Given a graph $G = (V, E)$ and a positive integer k, does there exist a subset $V' \subseteq V$ of size at most k such that for every edge $(u, v) \in E$ either $u \in V'$ or $v \in V'$.

Vertex Cover can be modelled as 2-HS with universe $U = V$ and $\mathcal{C} = \{\{u, v\} \mid (uv) \in E\}$ and hence using Theorem 1 we get a kernel with at most $4k^2$ edges and $8k^2$ vertices. Here we give a kernel with at most $4k$ vertices.

Now we define crown decomposition.

Definition 2. *A crown decomposition of a graph $G = (V, E)$ is a partitioning of V as C, H and R, where C and H are nonempty and the partition satisfies the following properties.*

1. *C is an independent set.*
2. *There are no edges between vertices of C and R, that is $N[C] \cap R = \emptyset$.*
3. *Let E' be the set of edges between vertices of C and H. Then E' contains a matching of size $|H|$, that is the bipartite subgraph $G' = (C \cup H, E')$ has a matching saturating all the vertices of H.*

We need the following lemma by Chor et. al. [7] which makes it possible to find a crown decomposition efficiently.

Lemma 2. *If a graph $G = (V, E)$ has an independent set $I \subseteq V$ such that $|N(I)| < |I|$, then a crown decomposition (C, H, R) of G such that $C \subseteq I$ can be found in time $O(m+n)$, given G and I.*

The crown-decomposition gives us a global method to reduce the instance size. Its importance is evident from the following simple lemma.

Lemma 3. *Let (C, H, R) be a crown decomposition of a graph $G = (V, E)$. Then G has a vertex cover of size k if and only if $G' = G[R]$ has a vertex cover of size $k' = k - |H|$.*

Proof. Suppose G has a vertex cover V' of size k in G. Now, we have a matching of size $|H|$ between C and H that saturates every vertex of H. Thus $|V' \cap (H \cup C)| \geq |H|$, as any vertex cover must pick one vertex form each of the matching edge. Hence the number of vertices in V' covering the edges not incident to $H \cup C$ is at most $k - |H|$, proving one direction of the result.

For the other direction, it is enough to observe that if V'' is a vertex cover of size $k - |H|$ for G' then $V'' \cup H$ is a vertex cover of size k for G. □

Theorem 2. *Vertex Cover has a kernel of size $4k$.*

Proof. Given an input graph $G = (V, E)$ and a positive integer k, we do as follows. We first find a maximal matching M of G. Let $V(M)$ be the set of endpoints of edges in M. Now if $|V(M)| > 2k$, we answer NO and stop as any vertex cover must contain at least one vertex from each of the matching edges and hence has size more than k. Now we distinguish two cases based on the size of $|V - V(M)|$. If $|V - V(M)| \leq 2k$, then we stop as we have obtained a kernel of size at most $4k$. Else $|V - V(M)| > 2k$. In this case we have found an independent set $I = V - V(M)$ such that $|N(I)| \leq |V(M)| < |I|$ and hence we can apply Lemma 2 to obtain a crown decomposition (C, H, R) of G. Given a crown decomposition (C, H, R), we apply Lemma 3 and obtain a smaller instance for a vertex cover with $G' = G[R]$ and parameter $k' = k - |H|$. Now we repeat the above procedure with this reduced instance until either we get a NO answer or we have $|V - V(M)| \leq 2k$ resulting in a kernel of size $4k$. □

The bound obtained on the kernel for Vertex Cover in Theorem 2 can be further improved to $2k$ with much more sophisticated use of crown decomposition. An alternate method to obtain a $2k$ size kernel for Vertex Cover is through a Linear Programming formulation of Vertex Cover. See [10] and [15] for further details on Linear Programming based kernelization of Vertex Cover.

3.4 Clique Cover

Unfortunately, not all known problem kernels are shown to have polynomial size. Here, we present some data reduction results with exponential-size kernels. Clearly, it is a pressing challenge to find out whether these bounds can be improved to polynomial ones.

In this section, we study the (EDGE) CLIQUE COVER problem, where the input consists of an undirected graph $G = (V, E)$ and a nonnegative integer k and the question is whether there is a set of at most k cliques in G such that each edge in E has both its endpoints in at least one of the selected cliques.

Given an n-vertex and m-edge graph G, we use $N(v)$ to denote the neighborhood of vertex v in G, namely, $N(v) := \{u \mid \{u, v\} \in E\}$. The *closed* neighborhood of vertex v, denoted by $N[v]$, is equal to $N(v) \cup \{v\}$.

We formulate data reduction rules for a generalized version of (EDGE) CLIQUE COVER in which already some edges may be marked as "covered". Then, the question is to find a clique cover of size k that covers all uncovered edges. We apply the following data reduction rules [14]:

1. Remove isolated vertices and vertices that are only adjacent to covered edges.
2. If an uncovered edge $\{u, v\}$ is contained in exactly one maximal clique C, that is, if the common neighbors of u and v induce a clique, then add C to the solution, mark its edges as covered, and decrease k by one.
3. If there is an edge $\{u, v\}$ whose endpoints have exactly the same closed neighborhood, that is, $N[u] = N[v]$, then mark all edges incident to u as covered. To reconstruct a solution for the non-reduced instance, add u to every clique containing v.

The correctness of the rules is easy to prove. To show the following problem kernel, only the first and third rule are needed.

Theorem 3 ([14])**.** (EDGE) CLIQUE COVER *admits a problem kernel with at most* 2^k *vertices.*

Proof. Consider any graph $G = (V, E)$ with more than 2^k vertices that has a clique cover $C_1, \ldots, C_k$ of size k. We assign to each vertex $v \in V$ a binary vector b_v of length k where bit i, $1 \leq i \leq k$, is set to 1 iff v is contained in clique C_i. Since there are only 2^k possible vectors, there must be $u \neq v \in V$ with $b_u = b_v$. If b_u and b_v are zero, the first rule applies; otherwise, u and v are contained in the same cliques. This means that u and v are connected and share the same neighborhood, and thus the third rule applies. □

4 Lower Bound Machinery

It is easy to see that if a decidable problem admits an $f(k)$ kernel for some function f, then the problem is FPT. Interestingly, the converse also holds, that is, if a problem is FPT then it admits an $f(k)$ kernel for some function f [15]. The proof of this fact is quite simple, and we present it here.

Fact 1 (Folklore, [15])**.** *If a parameterized problem* Π *is* FPT *then* Π *admits an* $f(k)$ *kernel for some function* f*.*

Proof. Suppose there is a decision algorithm for Π running in $f(k)n^c$ time for some function f and constant c. Given an instance (I, k) with $|I| = n$, if $n \geq f(k)$ then we run the decision algorithm on the instance in time $f(k)n^c \leq n^{c+1}$. If the decision algorithm outputs *yes*, the kernelization algorithm outputs a constant size *yes* instance, and if the decision algorithm outputs *no*, the kernelization algorithm outputs a constant size *no* instance. On the other hand, if $n < f(k)$ the kernelization algorithm just outputs (I, k). This yields an $f(k)$ kernel for the problem. □

Fact 1 implies that a problem has a kernel if and only if it is fixed parameter tractable. However, we are interested in kernels that are as small as possible, and a kernel obtained using Fact 1 has size that equals the dependence on k in the running time of the best known FPT algorithm for the problem. The question is - can we do better? The answer is that quite often we can as we saw in the previous section but many times we can not. It is only very recently that a methodology to rule out polynomial kernels has been developed [3, 12]. In this chapter we survey the tecniques that have been developed to show kernelization lower bounds. In this section we suvey some of the recently developed techniques for showing that problems do not admit polynomial kernels.

Consider the LONGEST PATH problem. It is well known that the LONGEST PATH problem can be solved in time $O(c^k n^{O(1)})$ using the well known method of COLOR-CODING. Is it feasible that it also admits a polynomial kernel? We argue that intuitively this should not be possible. Consider a large set $(G_1, k), (G_2, k), \ldots, (G_t, k)$ of instances to the LONGEST PATH problem. If we make a new graph G by just taking the disjoint union of the graphs $G_1 \ldots G_t$ we see that G contains a path of length k if and only if G_i contains a path of length k for some $i \leq t$. Suppose the LONGEST PATH problem had a polynomial kernel, and we ran the kernelization algorithm on G. Then this algorithm would in polynomial time return a new instance (G', k') such that $|V(G)| = k^{O(1)}$, a number potentially much smaller than t. This means that in some sense, the kernelization algorithm considers the instances $(G_1, k), (G_2, k) \ldots (G_t, k)$ and in *polynomial time* figures out which of the instances are the most likely to contain a path of length k. However, at least intuitively, this seems almost as difficult as solving the instances themselves and since the LONGEST PATH problem is NP-complete, this seems unlikely. We now formalize this intuition.

Definition 3. [Distillation [3]]

- *An OR-distillation algorithm for a language $L \subseteq \Sigma^*$ is an algorithm that receives as input a sequence $x_1, \ldots, x_t$, with $x_i \in \Sigma^*$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i|$, and outputs $y \in \Sigma^*$ with (a) $y \in L \iff x_i \in L$ for* some *$1 \leq i \leq t$ and (b) $|y|$ is polynomial in $\max_{i \leq t} |x_i|$. A language L is OR-distillable if there is a OR-distillation algorithm for it.*

- *An AND-distillation algorithm for a language $L \subseteq \Sigma^*$ is an algorithm that receives as input a sequence $x_1, \ldots, x_t$, with $x_i \in \Sigma^*$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i|$, and outputs $y \in \Sigma^*$ with (a) $y \in L \iff x_i \in L$ for* all *$1 \leq i \leq t$ and (b) $|y|$ is polynomial in $\max_{i \leq t} |x_i|$. A language L is AND-distillable if there is an AND-distillation algorithm for it.*

Observe that the notion of distillation is defined for unparameterized problems. Bodlaender et al. [3] conjectured that no NP-complete language can have an OR-distillation or an AND-distillation algorithm.

Conjecture 1 (OR-Distillation Conjecture [3])**.** *No* NP*-complete language L is OR-distillable.*

Conjecture 2 (AND-Distillation Conjecture [3])**.** *No* NP*-complete language L is AND-distillable.*

One should notice that if any NP-complete language is distillable, then so are all of them. Fortnow and Santhanam [12] were able to connect the OR-Distillation Conjecture to a well-known conjecture in classical complexity. In particular they proved that if the OR-Distillation Conjecture fails, the *polynomial time hierarchy* [17] collapses to the third level, a collapse that is deemed unlikely. No such connection is currently known for the AND-Distillation Conjecture, and for reasons soon to become apparent, a proof of such a connection would have significant impact in Parameterized Complexity. By PH=Σ_p^3 we will denote the complexity-theoretic event that the polynomial time hierarchy collapses to the third level.

Theorem 4 ([12])**.** *If the OR-Distillation Conjecture fails, then* PH=Σ_p^3*.*

We are now ready to define the parameterized analogue of distillation algorithms and connect this notion to the Conjectures 1 and 2

Definition 4. [Composition [3]]

- *A composition algorithm (also called OR-composition algorithm) for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), \ldots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^t |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}^+$ with (a) $(y, k') \in \Pi \iff (x_i, k) \in \Pi$ for* some *$1 \leq i \leq t$ and (b) k' is polynomial in k. A parameterized problem is compositional (or* OR-compositional*) if there is a composition algorithm for it.*
- *An AND-composition algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), \ldots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}^+$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^t |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}^+$ with (a) $(y, k') \in \Pi \iff (x_i, k) \in \Pi$ for* all *$1 \leq i \leq t$ and (b) k' is polynomial in k. A parameterized problem is AND-compositional if there is an AND-composition algorithm for it.*

Composition and distillation algorithms are very similar. The main difference between the two notions is that the restriction on output size for distillation algorithms is replaced by a restriction on the parameter size for the instance the composition algorithm outputs. We define the notion of the *unparameterized version* of a parameterized problem L. The mapping of parameterized problems to unparameterized problems is done by mapping (x, k) to the string $x\#1^k$, where $\# \notin \Sigma$ denotes the blank letter and 1 is an arbitrary letter in Σ. In this way, the unparameterized version of a parameterized problem Π is the language $\tilde{\Pi} = \{x\#1^k \mid (x, k) \in \Pi\}$. The following theorem yields the desired connection between the two notions.

Theorem 5 ([3, 12]). *Let Π be a compositional parameterized problem whose unparameterized version $\widetilde{\Pi}$ is* NP*-complete. Then, if Π has a polynomial kernel then* PH=Σ_p^3. *Similarly, let Π be an AND-compositional parameterized problem whose unparameterized version $\widetilde{\Pi}$ is* NP*-complete. Then, if Π has a polynomial kernel the AND-Distillation Conjecture fails.*

We can now formalize the discussion from the beginning of this section.

Theorem 6 ([3]). Longest Path *does not admit a polynomial kernel unless* PH=Σ_p^3.

Proof. The unparameterized version of Longest Path is known to be NP-complete [13]. We now give a composition algorithm for the problem. Given a sequence $(G_1, k) \ldots (G_t, k)$ of instances we output (G, k) where G is the disjoint union of $G_1 \ldots G_t$. Clearly G contains a path of length k if and only if G_i contains a path of length k for some $i \leq t$. By Theorem 5 Longest Path does not have a polynomial kernel unless PH=Σ_p^3. □

An identical proof can be used to show that the Longest Cycle problem does not admit a polynomial kernel unless PH=Σ_p^3. For many problems, it is easy to give AND-composition algorithms. For instance, the "disjoint union" trick yields AND-composition algorithms for the Treewidth, Cutwidth and Pathwidth problems, among many others. Coupled with Theorem 5 this implies that these problems do not admit polynomial kernels unless the AND-Distillation Conjecture fails. However, to this date, there is no strong complexity theoretic evidence known to support the AND-Distillation Conjecture. Therefore it would be interesting to see if such evidence could be provided.

For some problems, obtaining a composition algorithm directly is a difficult task. Instead, we can give a reduction from a problem that provably has no polynomial kernel unless PH=Σ_p^3 to the problem in question such that a polynomial kernel for the problem considered would give a kernel for the problem we reduced from. We now define the notion of *polynomial parameter transformations.*

Definition 5 ([4]). *Let P and Q be parameterized problems. We say that P is polynomial parameter reducible to Q, written $P \leq_{Ptp} Q$, if there exists a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N}$ and a polynomial p, such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ (a) $(x, k) \in P$ if and only $(x', k') = f(x, k) \in Q$ and (b) $k' \leq p(k)$. The function f is called* polynomial parameter transformation.

Proposition 1 ([4]). *Let P and Q be the parameterized problems and $\tilde{P}$ and $\tilde{Q}$ be the unparameterized versions of P and Q respectively. Suppose that $\tilde{P}$ is* NP*-complete and $\tilde{Q}$ is in* NP. *Furthermore if there is a polynomial parameter transformation from P to Q, then if Q has a polynomial kernel then P also has a polynomial kernel.*

Proposition 1 shows how to use polynomial parameter transformations to show kernelization lower bounds. A notion similar to polynomial parameter transformation was independently used by Fernau et al. [9] albeit without being explicitly defined. We now give an example of how Proposition 1 can be useful for showing that a problem does not admit a polynomial kernel. In particular, we show that the Path Packing problem does not admit a polynomial kernel unless PH=Σ_p^3. In this problem you are given a graph G together with an integer k and asked whether there exists a collection of k mutually vertex-disjoint paths of length k in G. This problem is known to be fixed parameter tractable [2] and is easy to see that for this problem the "disjoint union" trick discussed earlier does not directly apply. Thus we resort to polynomial parameter transformations.

Theorem 7. PATH PACKING *does not admit a polynomial kernel unless* PH=Σ_p^3.

Proof. We give a polynomial parameter transformation from the LONGEST PATH problem. Given an instance (G, k) to LONGEST PATH we construct a graph G' from G by adding $k-1$ vertex disjoint paths of length k. Now G contains a path of length k if and only if G' contains k paths of length k. This concludes the proof. □

Now we give several non-trivial applications of this framework developed by Bodlaender et al. [3] and Fortnow and Santhanam [12]. In particular we describe a "cookbook" for showing kernelization lower bounds using explicit identification. We then apply this cookbook to show that a wide variety of problems do not admit polynomial kernels. To show that a problem does not admit a polynomial size kernel we go through the following steps.

1. Find a suitable parameterization of the problem considered. Quite often parameterizations that impose extra structure make it easier to give a composition algorithm.
2. Define a suitable colored version of the problem. This is in order to get more control over how solutions to problem instances can look.
3. Show that the colored version of the problem is NP-complete.
4. Give a polynomial parameter transformation from the colored to the uncolored version. This will imply that if the uncolored version has a polynomial kernel then so does the colored version. Hence kernelization lower bounds for the colored version directly transfer to the original problem.
5. Show that the colored version parameterized by k is solvable in time $2^{k^c} \cdot n^{O(1)}$ for a fixed constant c.
6. Finally, show that the colored version is compositional and thus has no polynomial kernel. To do so, proceed as follows.
   (a) If the number of instances in the input to the composition algorithm is at least 2^{k^c} then running the parameterized algorithm on each instance takes time polynomial in input size. This automatically yields a composition algorithm.
   (b) If the number of instances is less than 2^{k^c}, every instance receives an unique identifier. Notice that in order to uniquely code the identifiers (ID) of all instances, k^c bits per instance is sufficient. The IDs are coded either as an integer, or as a subset of a $poly(k)$ sized set.
   (c) Use the coding power provided by colors and IDs to complete the composition algorithm.

In the following sections we show how to apply this approach to show incompressibility and kernelization lower bounds for a variety of problems.

4.1 Steiner Tree, Variants of Vertex Cover, and Bounded Rank Set Cover

The problems Steiner Tree, Connected Vertex Cover (ConVC), Capacitated Vertex Cover (CapVC), and Bounded Rank Set Cover are defined as follows. In Steiner Tree we are given a graph a graph $G = (T \cup N, E)$ and an integer k and asked for a vertex set $N' \subseteq N$ of size at most k such that $G[T \cup N']$ is connected. In ConVC we are given a graph $G = (V, E)$ and an integer k and asked for a vertex cover of size at most k that induces a connected subgraph in G. A *vertex cover* is a set $C \subseteq V$ such that each edge in E has at least one endpoint in C. The problem CapVC takes as input a graph $G = (V, E)$, a capacity function $cap : V \to \mathbb{N}^+$ and an integer k, and the task is to find a vertex cover C and a mapping from E to C in such a way that at most $cap(v)$ edges are mapped to every vertex $v \in C$. Finally, an instance of Bounded Rank Set Cover consists of a set family $\mathcal{F}$ over a universe U where every set $S \in \mathcal{F}$ has size at most d, and a positive integer k. The task is to find a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size at most k such that $\cup_{S \in \mathcal{F}'} S = U$. All four problems are known to be NP-complete (e.g., see [13] and the proof of Theorem 8); in this section, we show that the problems do not admit polynomial kernels for the parameter $(|T|, k)$ (in the case of Steiner Tree), k (in the case of ConVC and CapVC), and (d, k) (in the case of Bounded Rank Set Cover), respectively. To this end, we first use the framework presented at the beginning of this chapter to prove that another problem, which is called RBDS, does not have a polynomial kernel. Then, by giving polynomial parameter transformations from RBDS to the above problems, we show the non-existence of polynomial kernels for these problems.

In Red-Blue Dominating Set (RBDS) we are given a bipartite graph $G = (T \cup N, E)$ and an integer k and asked whether there exists a vertex set $N' \subseteq N$ of size at most k such that every vertex in T has at least one neighbor in N'. We show that RBDS parameterized by $(|T|, k)$ does not have a polynomial kernel. In the literature, the sets T and N are called "blue vertices" and "red vertices", respectively. In this paper we will call the vertices "terminals" and "nonterminals" in order to avoid confusion with the colored version of the problem that we are going to introduce. RBDS is equivalent to Set Cover and Hitting Set and is, therefore, NP-complete [13].

In the colored version of RBDS, denoted by Colored Red-Blue Dominating Set (Col-RBDS), the vertices of N are colored with colors chosen from $\{1, \ldots, k\}$, that is, we are additionally given a function $col \colon N \to \{1, \ldots, k\}$, and N' is required to contain exactly one vertex of each color. We will now follow the framework described at the beginning of this chapter.

Lemma 4. (1) *The unparameterized version of* Col-RBDS *is* NP*-complete.* (2) *There is a polynomial parameter transformation from* Col-RBDS *to* RBDS. (3) Col-RBDS *is solvable in* $2^{|T|+k} \cdot |T \cup N|^{O(1)}$ *time.*

Proof. (1) It is easy to see that Col-RBDS is in NP. To prove its NP-hardness, we reduce the NP-complete problem RBDS to Col-RBDS: Given an instance $(G = (T \cup N, E), k)$ of RBDS, we construct an instance $(G' = (T \cup N', E'), k, col)$ of Col-RBDS where the vertex set N' consists of k copies $v^1, \ldots, v^k$ of every vertex $v \in V$, one copy of each color. That is, $N' = \bigcup_{a \in \{1,\ldots,k\}} \{v^a \mid v \in N\}$, and the color of every vertex $v^a \in N_a$ is $col(v^a) = a$. The edge set E' is given by

$$E' = \bigcup_{a\in\{1,\dots,k\}} \{\{u, v^a\} \mid u \in T \wedge a \in \{1,\dots,k\} \wedge \{u, v\} \in E\}\,.$$

Now, there is a set $S \subset N$ of size k dominating all vertices in T in G if and only if in G', there is a set $S' \subset N'$ of size k containing one vertex of each color.

(2) Given an instance $(G = (T \cup N, E), k, col)$ of Col-RBDS, we construct an instance $(G' = (T' \cup N, E'), k)$ of RBDS. The set T' consists of all vertices from T plus k additional vertices $z_1, \dots, z_k$. The edge set E' consists of all edges from E plus the edges

$$\{\{z_a, v\} \mid a \in \{1,\dots,k\} \;\wedge\; v \in N \;\wedge\; col(v) = a\}.$$

Now, there is a set $S' \subset N'$ of size k dominating all vertices in T' in G' if and only if in G, there is a set $S \subset N$ of size k containing one vertex of each color.

(3) To solve Col-RBDS in the claimed running time, we first use the reduction given in (2) from Col-RBDS to RBDS. The number $|T'|$ of terminals in the constructed instance of RBDS is $|T|+k$. Next, we transform the RBDS instance (G', k) into an instance $(\mathcal{F}, U, k)$ of Set Cover where the elements in U one-to-one correspond to the vertices in T' and the sets in $\mathcal{F}$ one-to-one correspond to the vertices in N. Since Set Cover can be solved in $O(2^{|U|} \cdot |U| \cdot |\mathcal{F}|)$ time [11, Lemma 2], statement (3) follows.

□

Lemma 5. Col-RBDS *parameterized by* $(|T|, k)$ *is compositional.*

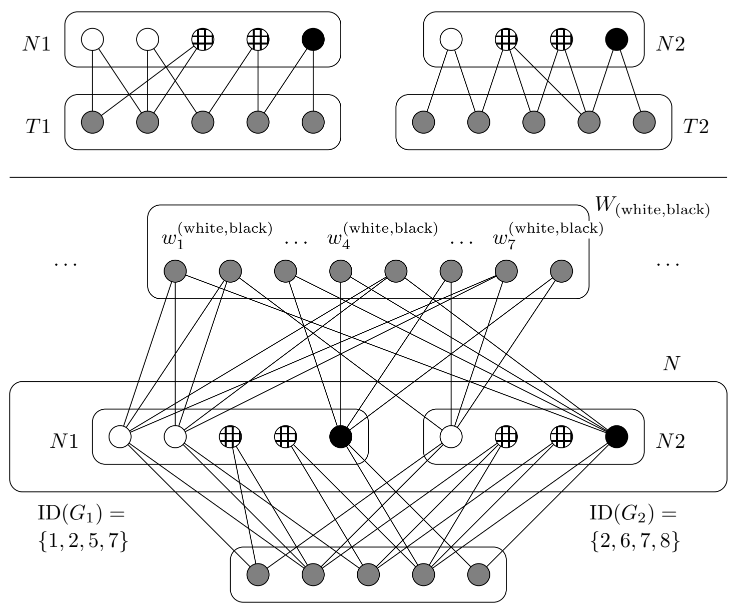
Proof. Given a sequence

$$(G_1 = (T_1 \cup N_1, E_1), k, col_1), \dots, (G_t = (T_t \cup N_t, E_t), k, col_t)$$

of Col-RBDS instances with $|T_1| = |T_2| = \ldots = |T_t| = p$, we show how to construct a Col-RBDS instance $(G = (T \cup N, E), k, col)$ as described in Definition 4.

For $i \in \{1,\dots,t\}$, let $T_i := \{u^i_1, \dots, u^i_p\}$ and $N_i := \{v^i_1, \dots, v^i_{q_i}\}$. We start with adding p vertices $u_1, \dots, u_p$ to the set T of terminals to be constructed. (We will add more vertices to T later.) Next, we add to the set N of nonterminals all vertices from the vertex sets $N_1, \dots, N_t$, preserving the colors of the vertices. That is, we set $N = \bigcup_{i\in\{1,\dots,t\}} N_i$, and for every vertex $v^i_j \in N$ we define $col(v^i_j) = col_i(v^i_j)$. Now, we add the edge set $\bigcup_{i\in\{1,\dots,t\}} \left\{\{u_{j_1}, v^i_{j_2}\} \mid \{u^i_{j_1}, v^i_{j_2}\} \in E_i\right\}$ to G (see Figure 1). The graph G and the coloring col constructed so far have the following property: If at least one of the Col-RBDS instances $(G_1, k, col_1), \dots, (G_t, k, col_t)$ is a *yes*-instance, then (G, k, col) is also a *yes*-instance because if for any $i \in \{1,\dots,t\}$ a size-k subset from N_i dominates all vertices in T_i, then the same vertex set selected from N also dominates all vertices in T. However, (G, k, col) may even be a *yes*-instance in the case where all instances $(G_1, k, col_1), \dots, (G_t, k, col_t)$ are *no*-instances, because in G one can select vertices into the solution that originate from different instances of the input sequence.

To ensure the correctness of the composition, we add more vertices and edges to G. We define for every graph G_i of the input sequence a unique identifier $\mathrm{ID}(G_i)$, which consists



$N1$
$T1$
$N2$
$T2$
$W_{(\text{white,black})}$
$w_1^{(\text{white,black})}$ ... $w_4^{(\text{white,black})}$ ... $w_7^{(\text{white,black})}$
...
...
N
$N1$
$N2$
$\text{ID}(G_1) = \{1,2,5,7\}$
$\text{ID}(G_2) = \{2,6,7,8\}$

Figure 1: Example for the composition algorithm for COL-RBDS. The upper part of the figure shows an input sequence consisting of two instances with $k = 3$ (there are three colors: white, checkered, and black). The lower part of the figure shows the output of the composition algorithm. For the sake of clarity, only the vertex set $W_{(\text{white,black})}$ is displayed, whereas five other vertex sets $W_{(a,b)}$ with $a, b \in \{\text{white, checkered, black}\}$ are omitted. Since $k = 3$ and $p = 5$, each ID should consist of eight numbers, and $W_{(\text{white,black})}$ should contain 16 vertices. For the sake of clarity, the displayed IDs consist of only four numbers each, and $W_{(\text{white,black})}$ contains only eight vertices.

of a size-$(p + k)$ subset of $\{1, \dots, 2(p + k)\}$ Since $\binom{2(p+k)}{p+k} \geq 2^{p+k}$ and since we can assume that the input sequence does not contain more than 2^{p+k} instances, it is always possible to assign unique identifiers to all instances of the input sequence. (Note that if there are more than 2^{p+k} instances, then we can solve all these instances in $\sum_{i=1}^{t} 2^{p+k} \cdot (p + q_i)^{O(1)} \leq t \cdot \sum_{i=1}^{t} (p + q_i)^{O(1)}$ time, which yields a composition algorithm.) For each color pair $(a, b) \in \{1, \dots, k\} \times \{1, \dots, k\}$ with $a \neq b$, we add a vertex set $W_{(a,b)} = \{w_1^{(a,b)}, \dots, w_{2(p+k)}^{(a,b)}\}$ to T, and we add to E the edge set

$$\bigcup_{i \in \{1,\dots,t\}, j_1 \in \{1,\dots,q_i\}} \Big\{ \{v_{j_1}^i, w_{j_2}^{(a,b)}\} \mid a = col(v_{j_1}^i) \wedge b \in \{1, \dots, k\} \setminus \{a\} \wedge j_2 \in \text{ID}(G_i) \Big\} \cup$$
$$\bigcup_{i \in \{1,\dots,t\}, j_1 \in \{1,\dots,q_i\}} \Big\{ \{v_{j_1}^i, w_{j_2}^{(a,b)}\} \mid b = col(v_{j_1}^i) \wedge a \in \{1, \dots, k\} \setminus \{b\} \wedge j_2 \notin \text{ID}(G_i) \Big\}$$

(see Figure 1).

Note that the construction conforms to the definition of a composition algorithm; in particular, k remains unchanged and the size of T is polynomial in p, k because $|T| = p + k(k-1) \cdot 2(p + k)$. To prove the correctness of the construction, we show that (G, k, col) has a solution $N' \subseteq N$ if and only if at least one instance (G_i, k, col_i) from the input sequence has a solution $N_i' \subseteq N_i$.

In one direction, if $N_i' \subseteq N_i$ is a solution for (G_i, k, col_i), then the same vertex set chosen from N forms a solution for (G, k, col). To see this, first note that the vertices from T are dominated by the chosen vertices. Moreover, for every color pair $(a, b) \in \{1, \ldots, k\} \times \{1, \ldots, k\}$ with $a \neq b$, each vertex from $W_{(a,b)}$ is either connected to all vertices v from N_i with $col(v) = a$ or to all vertices v from N_i with $col(v) = b$. Since N'_i contains one vertex of each color class from N_i, each vertex in $W_{(a,b)}$ is dominated by a vertex from N chosen into the solution.

In the other direction, to show that any solution $N' \subseteq N$ for (G, k, col) is a solution for at least one instance (G_i, k, col_i), we prove that N' cannot contain vertices originating from different instances of the input sequence. To this end, note that each two vertices in N' must have different colors. Assume, for the sake of a contradiction, that N' contains a vertex $v^{i_1}_{j_1}$ with $col(v^{i_1}_{j_1}) = a$ originating from the instance (G_{i_1}, k, col_{i_1}) and a vertex $v^{i_2}_{j_2}$ with $col(v^{i_2}_{j_2}) = b$ originating from a different instance (G_{i_2}, k, col_{i_2}). Due to the construction of the IDs, we have $\mathrm{ID}(G_{i_1}) \setminus \mathrm{ID}(G_{i_2}) \neq \emptyset$ and $\mathrm{ID}(G_{i_2}) \setminus \mathrm{ID}(G_{i_1}) \neq \emptyset$. This implies that there are vertices in $W_{(a,b)}$ (namely, all vertices $w^{(a,b)}_j$ with $j \in \mathrm{ID}(G_{i_2}) \setminus \mathrm{ID}(G_{i_1})$) and vertices in $W_{(b,a)}$ (namely, all vertices $w^{(b,a)}_j$ with $j \in \mathrm{ID}(G_{i_1}) \setminus \mathrm{ID}(G_{i_2})$) that are neither adjacent to $v^{i_1}_{j_1}$ nor to $v^{i_2}_{j_2}$. Therefore, N' does not dominate all vertices from T, which is a contradiction to the fact that N' is a solution for (G, k, col). $\square$

Theorem 8. ([8]) Red-Blue Dominating Set *and* Steiner Tree*, both parameterized by* $(|T|, k)$*,* Connected Vertex Cover *and* Capacitated Vertex Cover*, both parameterized by* k*, and* Bounded Rank Set Cover*, parameterized by* (k, d)*, do not admit polynomial kernels unless* $PH = \Sigma^3_p$*.*

Proof. For RBDS the statement of the theorem follows directly by Theorem 5 together with Lemmata 4 and 5.

To show that the statement is true for the other four problems, we give polynomial parameter transformations from RBDS to each of them—due to Proposition 1, this suffices to prove the statement. Let $(G = (T \cup N, E), k)$ be an instance of RBDS. To transform it into an instance $(G' = (T' \cup N, E'), k)$ of Steiner Tree, define $T' = T \cup \{\tilde{u}\}$ where $\tilde{u}$ is a new vertex and let $E' = E \cup \{\{\tilde{u}, v_i\} \mid v_i \in N\}$. It is easy to see that every solution for Steiner Tree on (G', k) one-to-one corresponds to a solution for RBDS on (G, k).

To transform (G, k) into an instance $(G'' = (V'', E''), k'')$ of ConVC, first construct the graph $G' = (T' \cup N, E')$ as described above. The graph G'' is then obtained from G' by attaching a leaf to every vertex in T'. Now, G'' has a connected vertex cover of size $k'' = |T'| + k = |T| + 1 + k$ iff G' has a steiner tree containing k vertices from N iff all vertices from T can be dominated in G by k vertices from N.

Next, we describe how to transform (G, k) into an instance $(G''' = (V''', E'''), cap, k''')$ of CapVC. First, for each vertex $u_i \in T$, add a clique to G''' that contains four vertices $u^0_i, u^1_i, u^2_i, u^3_i$. Second, for each vertex $v_i \in N$, add a vertex v'''_i to G'''. Finally, for each edge $\{u_i, v_j\} \in E$ with $u_i \in T$ and $v_j \in N$, add the edge $\{u^0_i, v'''_j\}$ to G'''. The capacities of the vertices are defined as follows: For each vertex $u_i \in T$, the vertices $u^1_i, u^2_i, u^3_i \in V'''$ have capacity 1 and the vertex $u^0_i \in V'''$ has capacity $\deg_{G'''}(u^0_i) - 1$. Each vertex v'''_i has capacity $\deg_{G'''}(v'''_i)$. Clearly, in order to cover the edges of the size-4 cliques inserted for the vertices of T, every capacitated vertex cover for G''' must contain all vertices $u^0_i, u^1_i, u^2_i, u^3_i$. Moreover, since the capacity of each vertex u^0_i is too small to cover all edges incident to u^0_i, at least one neighbor v_j''' of u_i^0 must be selected into every capacitated vertex cover for G'''. Therefore, it is not hard to see that G''' has a capacitated vertex cover of size $k''' = 4 \cdot |T| + k$ iff all vertices from T can be dominated in G by k vertices from N.

Finally, to transform (G, k) into an instance $(\mathcal{F}, U, k)$ of Bounded Rank Set Cover, add one element e_i to U for every vertex $u_i \in T$. For every vertex $v_j \in N$, add one set $\{e_i \mid \{u_i, v_j\} \in E\}$ to $\mathcal{F}$. The correctness of the construction is obvious, and since $|U| = |T|$, every set in $\mathcal{F}$ contains at most $d = |T|$ elements. $\square$

References

[1] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.

[2] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. Assoc. Comput. Mach.*, 42(4):844–856, 1995.

[3] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. In *Proc. 35th ICALP*, volume 5125 of *LNCS*, pages 563–574. Springer, 2008.

[4] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels. Technical Report UU-CS-2008-030, Department of Information and Computing Sciences, Utrecht University, 2008.

[5] Jianer Chen, Henning Fernau, Iyad A. Kanj, and Ge Xia. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. In *Proc. 22nd STACS*, volume 3404 of *LNCS*, pages 269–280. Springer, 2005.

[6] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex Cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

[7] Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in o(n^2) steps. In *Proc. 30th WG*, volume 3353 of *LNCS*, pages 257–269. Springer, 2004.

[8] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through Colors and IDs. In *Proc. 36th ICALP*, To appear.

[9] Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Daniel Raible, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. In *Proc. 26th STACS*, pages 421–432.

[10] J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[11] Fedor V. Fomin, Dieter Kratsch, and Gerhard J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *Proc. 30th WG*, volume 3353 of *LNCS*, pages 245–256. Springer, 2004.

[12] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proc. 40th STOC*, pages 133–142. ACM Press, 2008.

[13] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[14] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction, exact, and heuristic algorithms for Clique Cover. In *Proc. 8th ACM-SIAM ALENEX*, pages 86–94. ACM-SIAM, 2006. Long version to appear in *The ACM Journal of Experimental Algorithmics*.

[15] Rolf Niedermeier. *An Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[16] W. V. Quine. The problem of simplifying truth functions. *Amer. Math. Monthly*, 59:521–531, 1952.

[17] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.

[18] Stéphan Thomassé. A quadratic kernel for feedback vertex set. In *Proc. 20th SODA*, pages 115–119. ACM/SIAM, 2009.