Cross-Composition: A New Technique for Kernelization Lower Bounds*

Hans L. Bodlaender[1], Bart M. P. Jansen[1], and Stefan Kratsch[1]

1 **Department of Information and Computing Sciences, Utrecht University**
P.O. Box 80.089, 3508 TB, Utrecht, The Netherlands
{hansb,bart,kratsch}@cs.uu.nl

Abstract

We introduce a new technique for proving kernelization lower bounds, called *cross-composition*. A classical problem L cross-composes into a parameterized problem Q if an instance of Q with polynomially bounded parameter value can express the logical OR of a sequence of instances of L. Building on work by Bodlaender et al. (ICALP 2008) and using a result by Fortnow and Santhanam (STOC 2008) we show that if an NP-hard problem cross-composes into a parameterized problem Q then Q does not admit a polynomial kernel unless the polynomial hierarchy collapses.

Our technique generalizes and strengthens the recent techniques of using OR-composition algorithms and of transferring the lower bounds via polynomial parameter transformations. We show its applicability by proving kernelization lower bounds for a number of important graphs problems with structural (non-standard) parameterizations, e.g., Chromatic Number, Clique, and Weighted Feedback Vertex Set do not admit polynomial kernels with respect to the vertex cover number of the input graphs unless the polynomial hierarchy collapses, contrasting the fact that these problems are trivially fixed-parameter tractable for this parameter. We have similar lower bounds for Feedback Vertex Set.

1998 ACM Subject Classification F.2.2

Keywords and phrases kernelization, lower bounds, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2011.165

1 Introduction

Preprocessing and data reduction are important and widely applied concepts for speeding up polynomial-time algorithms or for making computation feasible at all in the case of hard problems that are not believed to have efficient algorithms. Kernelization is a way of formalizing data reduction, which allows for a formal analysis of the (im)possibility of data reduction and preprocessing. It originated as a technique to obtain fixed-parameter tractable algorithms for hard (parameterized) problems, and has evolved into its own topic of research (see [19, 2] for recent surveys). A *parameterized problem* [14, 16] is a language $Q \subseteq \Sigma^* \times \mathbb{N}$, the second component is called the *parameter*. A *kernelization* algorithm (*kernel*) transforms an instance (x, k) in polynomial time into an equivalent instance (x', k') such that $|x'|, k' \leq f(k)$ for some computable function f, which is the *size* of the kernel.

From a practical perspective we are particularly interested in cases where $f \in k^{O(1)}$, so-called *polynomial kernels*. Success stories of kernelization include the $O(k^2)$ kernel for k-Vertex Cover containing at most $2k$ vertices [11] and the meta-theorems for kernelization

* This work was supported by the Netherlands Organisation for Scientific Research (NWO), project "KERNELS: Combinatorial Analysis of Data Reduction".

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 165–176

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of problems on planar graphs [4], among many others (cf. also [22]). Although researchers have looked for polynomial kernels for elusive problems such as k-Path for many years, it was only recently that techniques were introduced which make it possible to prove (under some complexity-theoretic assumption) that a parameterized problem in FPT does not admit a polynomial kernel. Bodlaender et al. [3] introduced the concept of a or-*composition* algorithm as a tool to give super-polynomial lower bounds on kernel sizes. Consider some set S, and let $\text{or}(S)$ denote the set such that for any sequence $x^* := (x_1, \ldots, x_t)$ of instances of S we have $x^* \in \text{or}(S) \Leftrightarrow \bigvee_{i=1}^{t} x_i \in S$; then we could say that the language $\text{or}(S)$ expresses the or of instances of S. The approach taken in the original paper by Bodlaender et al. [3] uses a theorem by Fortnow and Santhanam [17] to show that if there is a polynomial-time or-composition algorithm that maps any sequence of instances $(x_1, k), (x_2, k), \ldots, (x_t, k)$ of some parameterized problem Q which all share the same parameter value to an instance (x^*, k^*) of Q which acts as the or of the inputs and $k^* \in k^{O(1)}$, then Q does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. This machinery made it possible to prove e.g. that k-Path and the Clique problem parameterized by the treewidth of the graph do not admit polynomial kernels unless NP $\subseteq$ coNP/poly[1]. The latter is deemed unlikely since it is known to imply a collapse of the polynomial hierarchy to its third level [24] (and further [8]).

It did not take long before the techniques of Bodlaender et al. were combined with the notion of a *polynomial parameter transformation* to also prove lower bounds for problems for which no direct or-composition algorithm could be found. This idea was used implicitly by Fernau et al. [15] to show that k-Leaf Out-Branching does not admit a polynomial kernel, and was formalized in a paper by Bodlaender et al. [7]: they showed that if there is a polynomial-time transformation from P to Q which incurs only a polynomial blow-up in the parameter size, then if P does not admit a polynomial kernel then Q does not admit one either. These polynomial parameter transformations were used extensively by Dom et al. [13] who proved kernelization lower bounds for a multitude of important parameterized problems such as Small Universe Hitting Set and Small Universe Set Cover. Dell and van Melkebeek [12] were able to extend the techniques of Fortnow and Santhanam to prove, e.g., that Vertex Cover does not admit a kernel of size $O(k^{2-\epsilon})$ for any $\epsilon > 0$.

Our results. We introduce a new technique to prove kernelization lower-bounds, which we call *cross-composition*. This technique generalizes and strengthens the earlier methods of or-composition [3] and polynomial-parameter transformations [7], and puts the two existing methods of showing kernelization lower bounds in a common perspective. Whereas the existing notion of or-composition works by composing multiple instances of a *parameterized* problem Q into a single instance of Q with a bounded parameter value, for our new technique it is sufficient to compose the or of any *classical* NP-hard problem into an instance of the parameterized problem Q for which we want to prove a lower-bound. The term *cross* in the name stems from this fact: the source- and target problem of the composition need no longer be the same. Since the input to a cross-composition algorithm is a list of *classical* instances instead of parameterized instances, the inputs do not have a parameter in which the output parameter of the composition must be bounded; instead we require that the size of the output parameter is polynomially bounded in the size of the largest input instance. In addition we show that the output parameter may depend polynomially on the logarithm of the number of input instances, which often simplifies the constructions and proofs. We also introduce the concept of a *polynomial equivalence relation* to remove the need for padding

[1] In the remainder of this introduction we assume that NP $\not\subseteq$ coNP/poly when stating kernelization lower bounds.

Problem name	Parameter	Kernel size	
Clique	vertex cover	not polynomial	[Section 4.1]
Chromatic Number	vertex cover	not polynomial	[Section 4.2]
Feedback Vertex Set	dist. from cluster	not polynomial	[Section 4.3]
Feedback Vertex Set	dist. from co-cluster	not polynomial	[Section 4.3]
Weighted FVS	vertex cover	not polynomial	[Section 4.3]

Table 1 An overview of the kernelization lower bounds obtained in this paper; all listed problems are fixed-parameter tractable with respect to this parameterization. Section 4 describes the parameterized problems in more detail.

arguments which were frequently required for or-compositions.

To show the power of cross-composition we give kernelization lower bounds for *structural* parameterizations of several important graph problems. Since many combinatorial problems are easy on graphs of bounded treewidth [6], and since the treewidth of a graph is bounded by the vertex cover number, it is often thought that almost all problems become tractable when parameterized by the vertex cover number of the graph. We show that this is not the case for kernelization: Clique, Chromatic Number and Weighted Feedback Vertex Set do not admit polynomial kernels parameterized by the vertex cover number of the graph. In the case of Clique it was already known [3] that the problem does not admit a polynomial kernel parameterized by the treewidth of the graph; since the vertex cover number is at least as large as the treewidth we prove a stronger result. For the unweighted Feedback Vertex Set problem, which admits a polynomial kernel parameterized by the target size of the feedback set [23], we show that there is no polynomial kernel for the parameterization by deletion distance to cluster graphs or co-cluster graphs.

Organization. The paper is organized as follows. We first give some preliminary definitions. Section 3 gives the formal definition of cross-composition, and proves that cross-compositions allow us to give kernelization lower bounds. In Section 4 we apply the new technique to obtain kernelization lower bounds for various problems.

2 Preliminaries

In this work we only consider undirected, finite, simple graphs. Let G be a graph and denote its vertex set by $V(G)$ and the edge set by $E(G)$. We use $\chi(G)$ to denote the chromatic number of G. If $V' \subseteq V(G)$ then $G[V']$ denotes the subgraph of G induced by V'. A graph is a *cluster graph* if every connected component is a clique. A graph is a *co-cluster* graph if it is the edge-complement of a cluster graph. Throughout this work we use Σ to denote a finite alphabet, but note that multiple occurrences of Σ may refer to different alphabets. For positive integers n we define $[n] := \{1, \ldots, n\}$. The satisfiability problem for boolean formulae is referred to as sat. Several proofs have been deferred to the full version [5] of this paper due to space restrictions. For completeness we give the following core definitions of parameterized complexity [3, 14].

▶ **Definition 1.** A *parameterized problem* is a language $Q \subseteq \Sigma^* \times \mathbb{N}$, and is contained in the class (strongly uniform) FPT (for Fixed-Parameter Tractable) if there is an algorithm that decides whether $(x, k) \in Q$ in $f(k)|x|^{O(1)}$ time for some computable function f.

▶ **Definition 2.** A *kernelization* algorithm [19, 2], 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 $p(|x| + k)$ time a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that:

- $(x, k) \in Q \Leftrightarrow (x', k') \in Q$,
- $|x'|, k' \leq f(k)$,

where f is a computable function, and p a polynomial. Any function f as above is referred to as the size of the kernel; if f is a polynomial then we have a *polynomial kernel*.

3 Cross-Composition

3.1 The Definition

In this section we define the concept of cross-composition and give all the terminology needed to apply the technique.

▶ **Definition 3** (Polynomial equivalence relation)**.** An equivalence relation $\mathcal{R}$ on Σ^* is called a *polynomial equivalence relation* if the following two conditions hold:

1. There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether x and y belong to the same equivalence class in $(|x| + |y|)^{O(1)}$ time.
2. For any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into at most $(\max_{x \in S} |x|)^{O(1)}$ classes.

▶ **Definition 4** (Cross-composition)**.** Let $L \subseteq \Sigma^*$ be a set and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L *cross-composes* into Q if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which, given t strings $x_1, x_2, \ldots, x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that:

1. $(x^*, k^*) \in Q \Leftrightarrow x_i \in L$ for some $1 \leq i \leq t$,
2. k^* is bounded by a polynomial in $\max_{i=1}^{t} |x_i| + \log t$.

3.2 How Cross-compositions Yield Lower Bounds

The purpose of this section is to prove that cross-compositions yield kernelization lower bounds. To give this proof we need some concepts from earlier work [3, 17, 12].

▶ **Definition 5** ([17])**.** A *weak distillation of* SAT into a set $L \subseteq \Sigma^*$ is an algorithm that:

- receives as input a sequence $(x_1, \ldots, x_t)$ of instances of SAT,
- uses time polynomial in $\sum_{i=1}^{t} |x_i|$,
- and outputs a string $y \in \Sigma^*$ with

1. $y \in L \Leftrightarrow x_i \in$ SAT for some $1 \leq i \leq t$,
2. $|y|$ is bounded by a polynomial in $\max_{i=1}^{t} |x_i|$.

▶ **Theorem 6** (Theorem 1.2 [17])**.** *If there is a weak distillation of* SAT *into any set $L \subseteq \Sigma^*$ then NP $\subseteq$ coNP/poly and the polynomial-time hierarchy collapses to the third level ($PH = \Sigma_3^p$).*

▶ **Definition 7** ([12])**.** The OR *of a language $L \subseteq \Sigma^*$* is the set $\text{OR}(L)$ that consists of all tuples $(x_1, \ldots, x_t)$ for which there is an index $1 \leq i \leq t$ with $x_i \in L$.

▶ **Definition 8** ([3])**.** We associate an instance (x, k) of a parameterized problem with the *unparameterized instance* formed by the string $x\#1^k$, where $\#$ denotes a new character that we add to the alphabet and 1 is an arbitrary letter in Σ. The *unparameterized version* of a parameterized problem Q is the language $\tilde{Q} = \{x\#1^k \mid (x, k) \in Q\}$.

▶ **Theorem 9.** *Let $L \subseteq \Sigma^*$ be a set which is NP-hard under Karp reductions. If L cross-composes into the parameterized problem Q and Q has a polynomial kernel then there is a weak distillation of* SAT *into* OR$(\tilde{Q})$ *and NP $\subseteq$ coNP/poly.*

Proof. The proof is by construction and generalizes the concepts of Bodlaender et al. [3]. Assuming the conditions in the statement of the theorem hold, we show how to build an algorithm which distills SAT into OR$(\tilde{Q})$. By the definition of cross-composition there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm C which composes L-instances belonging to the same class of $\mathcal{R}$ into a Q-instance.

The input to the distillation algorithm consists of a sequence $(x_1, \ldots, x_t)$ of instances of SAT, which we may assume are elements of Σ^*. Define $m := \max_{j=1}^{t} |x_j|$. If $t > (|\Sigma|+1)^m$ then there must be duplicate inputs, since the number of distinct inputs of length $m' \leq m$ is $|\Sigma|^{m'}$. By discarding duplicates we may therefore assume that $t \leq (|\Sigma|+1)^m$, i.e., $\log t \in O(m)$. By the assumption that L is NP-hard under Karp reductions, there is a polynomial-time reduction from SAT to L. We use this reduction to transform each SAT instance x_i for $1 \leq i \leq t$ into an equivalent L-instance y_i. Since the transformation takes polynomial time, it cannot increase the size of an instance by more than a polynomial factor and therefore $|y_i|$ is polynomial in m for all i.

The algorithm now pairwise compares instances using the polynomial-time equivalence test of $\mathcal{R}$ (whose existence is guaranteed by Definition 3) to partition the L-instances $(y_1, \ldots, y_t)$ into partite sets $Y_1, \ldots, Y_r$ such that all instances from the same partite set are equivalent under $\mathcal{R}$. The properties of a polynomial equivalence relation guarantee that r is polynomial in m and that this partitioning step takes polynomial time in the total input size.

We now use the cross-composition algorithm C on each of the partite sets $Y_1, \ldots, Y_r$, which is possible since all instances from the same set are equivalent under $\mathcal{R}$. Let (z_i, k_i) be the result of applying C to a sequence containing the contents of the set Y_i, for $1 \leq i \leq r$. From the definition of cross-composition and using $\log t \in O(m)$ it follows that each k_i is polynomial in m, and that the computation of these parameterized instances takes polynomial time in the total input size. From Definition 4 it follows that (z_i, k_i) is a YES instance of Q if and only if one of the instances in Y_i is a YES instance of L, which in turn happens if and only if one of the inputs x_i is a YES instance of SAT.

Let K be a polynomial kernelization algorithm for Q, whose existence we assumed in the statement of the theorem. We apply K to the instance (z_i, k_i) to obtain an equivalent instance (z_i', k_i') of Q for each $1 \leq i \leq r$. Since K is a polynomial kernelization we know that these transformations can be carried out in polynomial time and that $|z_i'|, k_i' \leq k_i^{O(1)}$. Since k_i is polynomial in m it follows that $|z_i'|$ and k_i' are also polynomial in m for $1 \leq i \leq r$.

As the next step we convert each parameterized instance (z_i', k_i') to the unparameterized variant $\tilde{z}_i := z_i' \# 1^{k_i'}$. Since the values of the parameters are polynomial in m this transformation takes polynomial time, and afterwards we find that $|\tilde{z}_i|$ is polynomial in m for each $1 \leq i \leq r$.

The last stage of the algorithm simply combines all unparameterized variants into one tuple $x^* := (\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_r)$. Since the size of each component is polynomial in m, and since the number of components r is polynomial in m, we have that $|x^*|$ is polynomial in m. The tuple x^* forms an instance of OR$(\tilde{Q})$, and by the definition of OR$(\tilde{Q})$ we know that $x^* \in$ OR$(\tilde{Q})$ if and only if some element of the tuple is contained in $\tilde{Q}$. By tracing back the series of equivalences we therefore find that $x^* \in$ OR$(\tilde{Q})$ if and only if some input x_i is a YES-instance of SAT. Since we can construct x^* in polynomial time and $|x^*|$ is polynomial in m, we have constructed a weak distillation of SAT into OR$(\tilde{Q})$. By Theorem 6 this implies NP $\subseteq$ coNP/poly and proves the theorem. ◀

▶ **Corollary 10.** *If some set L is NP-hard under Karp reductions and L cross-composes into the parameterized problem Q then there is no polynomial kernel for Q unless NP $\subseteq$ coNP/poly.*

A simple extension of Theorem 9 shows that cross-compositions also exclude the possibility of compression into a small instance of a different parameterized problem, a notion sometimes referred to as bikernelization [20, 21]. If an NP-hard set cross-composes into a parameterized problem Q, then unless NP $\subseteq$ coNP/poly there is no polynomial-time algorithm that maps an instance (x, k) of Q to an equivalent instance (x', k') of *any* parameterized problem P with $|x'|, k' \leq k^{O(1)}$.

4 Results Based on Cross-Composition

In this section we apply the cross-composition technique to give kernelization lower bounds. We consider the problems Feedback Vertex Set, Chromatic Number and Clique under various parameterizations. The first parameter we consider is the vertex cover number of a graph G, i.e. the cardinality of a smallest set of vertices $Z \subseteq V(G)$ such that all edges of G have at least one endpoint in Z. We show that Clique, Chromatic Number and Weighted Feedback Vertex Set do *not* admit polynomial kernels parameterized by the size of a vertex cover unless NP $\subseteq$ coNP/poly.

We could also define the vertex cover number as the minimum number of vertex deletions needed to reduce a graph to an edgeless graph; hence the vertex cover number measures how far a graph is from being edgeless. Following the initiative of Cai [9] we may similarly define the deletion distance of a graph G to a (co-)cluster graph as the minimum number of vertices that have to be deleted from G to turn it into a (co-)cluster graph. Since (co-)cluster graphs have a very restricted structure, one would expect that a parameterization by (co-) cluster deletion distance leads to fixed-parameter tractability; indeed this is the case for many problems, since graphs of bounded (co-)cluster deletion distance also have bounded cliquewidth [1]. For the Feedback Vertex Set problem, which admits a polynomial kernel parameterized by the target size and hence by the vertex cover number, we show that the parameterizations by cluster deletion or co-cluster deletion distance do not admit polynomial kernels.

In Table 2 we give the known results for our subject problems with respect to the standard parameterization, which refers to the solution size. Since the problems we study are very well-known, we do not give a full definition for each one. Instead we give an educative example of how the parameter is reflected in an instance.

Chromatic Number parameterized by the size of a vertex cover
Instance: A graph G, a vertex cover $Z \subseteq V(G)$, and a positive integer ℓ.
Parameter: The size $k := |Z|$ of the vertex cover.
Question: Is $\chi(G) \leq \ell$, i.e., can G be colored with at most ℓ colors?

For technical reasons we supply a vertex cover in the input of the problem, to ensure that well-formed instances can be recognized in polynomial time. The parameter to the problem claims a bound on the vertex cover number of the graph, and using the set Z we may verify this bound. For Feedback Vertex Set parameterized by deletion distance to cluster graphs or co-cluster graphs, we also supply the deletion set in the input. These versions of the problem are certainly no harder to kernelize than the versions where a deletion set or vertex cover is not given.

Problem name	Parameter	Param. complexity	Kernel size
Clique	clique	W[1]-hard [14]	W[1]-hard [14]
Feedback Vertex Set	feedback vertex set	FPT [10]	$4k^2$ vertices [23]
Chromatic Number	chromatic number	NP-h for $k \in O(1)$	NP-h for $k \in O(1)$

Table 2 Parameterized complexity and kernel size for some of the problems considered in this paper, with respect to the standard parameterization (i.e., target size).

4.1 Clique parameterized by Vertex Cover

An instance of the NP-complete Clique problem [18, GT19] is a tuple (G, ℓ) and asks whether the graph G contains a clique on ℓ vertices. We use this problem for our first kernelization lower bound.

▶ **Theorem 11.** Clique *parameterized by the size of a vertex cover does not admit a polynomial kernel unless NP* $\subseteq$ *coNP/poly.*

Proof. We prove the theorem by showing that Clique cross-composes into Clique parameterized by vertex cover; by Corollary 10 this is sufficient to establish the claim. We define a polynomial equivalence relation $\mathcal{R}$ such that all bitstrings which do not encode a valid instance of Clique are equivalent, and two well-formed instances (G_1, ℓ_1) and (G_2, ℓ_2) are equivalent if and only if they satisfy $|V(G_1)| = |V(G_2)|$ and $\ell_1 = \ell_2$. From this definition it follows that any set of well-formed instances on at most n vertices each is partitioned into $O(n^2)$ equivalence classes. Since all malformed instances are in one class, this proves that $\mathcal{R}$ is indeed a polynomial equivalence relation.

We now give a cross-composition algorithm which composes t input instances $x_1, \ldots, x_t$ which are equivalent under $\mathcal{R}$ into a single instance of Clique parameterized by vertex cover. If the input instances are malformed or the size of the clique that is asked for exceeds the number of vertices in the graph, then we may output a single constant-size no instance; hence in the remainder we may assume that all inputs are well-formed and encode structures $(G_1, \ell), \ldots, (G_t, \ell)$ such that $|V(G_i)| = n$ for all $i \in [t]$ and all instances agree on the value of ℓ, which is at most n. We construct a single instance (G', Z', ℓ', k') of Clique parameterized by vertex cover, which consists of a graph G' with vertex cover $Z' \subseteq V(G')$ of size k' and an integer ℓ'.

Let the vertices in each G_i be numbered arbitrarily from 1 to n. We construct the graph G' as follows (see also Figure 1):

1. Create ℓn vertices $v_{i,j}$ with $i \in [\ell]$ and $j \in [n]$. Connect two vertices $v_{i,j}$ and $v_{i',j'}$ if $i \neq i'$ and $j \neq j'$. Let C denote the set of these vertices. It is crucial that any clique in G' can only contain one vertex $v_{i,\cdot}$ or $v_{\cdot,j}$ for each choice of $i \in [\ell]$ respectively $j \in [n]$. Thus any clique contains at most ℓ vertices from C.
2. For each pair $1 \leq p < q \leq n$ of distinct vertices from $[n]$ (i.e., vertices of graphs G_i), create three vertices: $w_{p,q}$, $w_{p,\hat{q}}$, and $w_{\hat{p},q}$ and make them adjacent to C as follows:
   a. $w_{p,q}$ is adjacent to all vertices from C,
   b. $w_{p,\hat{q}}$ is adjacent to all vertices from C except for $v_{\cdot,j}$ with $j = q$, and
   c. $w_{\hat{p},q}$ is adjacent to all vertices from C except for $v_{\cdot,j}$ with $j = p$.

   Furthermore we add all edges between vertices $w_{\cdot,\cdot}$ that correspond to distinct pairs from $[n]$. Let D denote these $3\binom{n}{2}$ vertices. Any clique can contain at most one $w_{\cdot,\cdot}$ vertex for each pair from $[n]$.

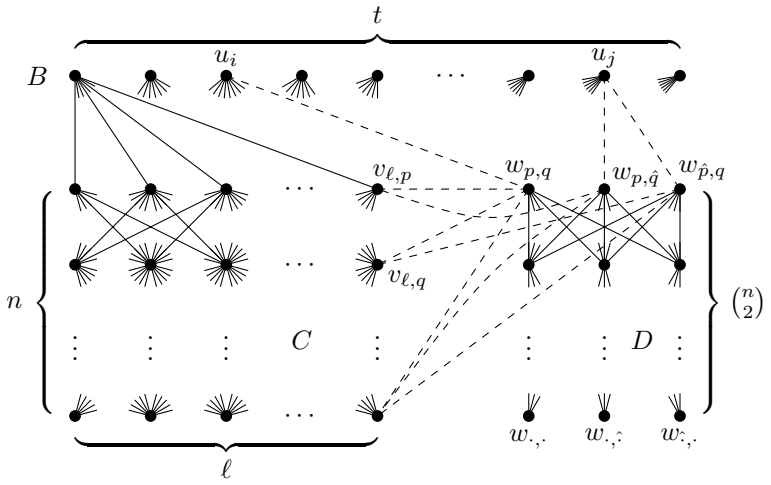

Figure 1 A sketch of the construction used in the proof of Theorem 11. The dashed edges show in an examplary way how vertices $w_{p,q}$, $w_{p,\hat{q}}$, and $w_{\hat{p},q}$ are connected to vertices of B and C, e.g., $\{p,q\}$ is an edge of G_i but not of G_j.

3. For each instance x_i with graph G_i make a new vertex u_i and connect it to all vertices in C. The adjacency to D is as follows:
   a. Make u_i adjacent to $w_{p,q}$ if $\{p,q\}$ is an edge in G_i.
   b. Otherwise make u_i adjacent to $w_{p,\hat{q}}$ and $w_{\hat{p},q}$.

   Let B denote this set of t vertices.

We define $\ell' := \ell + 1 + \binom{n}{2}$. Furthermore, we let $Z' := C \cup D$ which is easily verified to be a vertex cover for G' of size $k' := |Z'| = \ell n + 3\binom{n}{2}$. The value k' is the parameter to the problem, which is polynomial in n and hence in the size of the largest input instance. The cross-composition outputs the instance $x' := (G', Z', \ell', k')$. It is easy to see that our construction of G' can performed in polynomial time. Let us now argue that x' is YES if and only if at least one of the instances x_i is YES.

($\Leftarrow$) First we will assume that some x_{i^*} is YES, i.e., that G_{i^*} contains a clique on at least ℓ vertices. Let $S \subseteq [n]$ denote a clique of size exactly ℓ in G_{i^*}. We will construct a set S' of size $\ell' = \ell + 1 + \binom{n}{2}$ and show that it is a clique in G':

1. We add the vertex u_{i^*} to S'.
2. Let $S = \{p_1, \ldots, p_\ell\} \subseteq [n]$. For each p_j in S we add the vertex v_{j,p_j} to S'. By Step 1 all these vertices are pairwise adjacent, and by Step 3 they are adjacent to u_{i^*}.
3. For each pair $1 \leq p < q \leq n$ there are two cases:
   a. If $\{p,q\}$ is an edge of G_{i^*} then the vertex u_{i^*} is adjacent to $w_{p,q}$ in G' (by Step 3) and $w_{p,q}$ is adjacent to all vertices of C (by Step 2). We add $w_{p,q}$ to S'.
   b. Otherwise the vertex u_{i^*} is adjacent to both $w_{p,\hat{q}}$ and $w_{\hat{p},q}$. Since the clique S cannot contain both p and q when $\{p,q\}$ is a non-edge we are able to add $w_{p,\hat{q}}$ respectively $w_{\hat{p},q}$ to S'; recall that, e.g., $w_{p,\hat{q}}$ is adjacent to all vertices of C except those corresponding to q.

   In both cases we add one $w_{\cdot,\cdot}$-vertex to S', each corresponding to a different pair p,q; all these vertices are pairwise adjacent by Step 2.

We have identified the clique S' in G' of size $\ell' = \ell + 1 + \binom{n}{2}$, proving that x' is a YES-instance.

($\Rightarrow$) Now assume that x' is a YES-instance and let S' be a clique of size $\ell + 1 + \binom{n}{2}$ in G'. Since S' contains at most ℓ vertices from C (i.e., one $v_{i,\cdot}$ for each $i \in [\ell]$) and at most $\binom{n}{2}$ vertices from D it must contain at least one vertex from B, say $u_{i^*} \in B$. Since B is an independent set the set S' must contain exactly ℓ vertices from C and exactly $\binom{n}{2}$ vertices from D. Let $S = \{j \in [n] \mid v_{i,j} \in S' \text{ for some } i \in [\ell]\}$. The set S has size ℓ since S' contains at most one vertex $v_{\cdot,j}$ for each $j \in [n]$. We will now argue that S is a clique in G_{i^*}. Let $p, q \in S$. The clique S' must contain a $w_{\cdot,\cdot}$-vertex corresponding to $\{p, q\}$ and it must contain vertices $v_{i,p}$ and $v_{i',q}$ for some $i, i' \in [\ell]$. Therefore it must contain $w_{p,q}$ since $w_{p,\hat{q}}$ has no edges to vertices $v_{\cdot,q}$ and $w_{\hat{p},q}$ has no edges to $v_{\cdot,p}$ by Step 2. Thus $u_{i^*} \in S'$ must be adjacent to $w_{p,q}$ which implies that G_{i^*} contains the edge $\{p, q\}$. Thus S is a clique in G_{i^*}.

Since we proved that the instance (G', Z', ℓ', k') can be constructed in polynomial-time and that it acts as the OR of the input instances, and because the parameter value k' is bounded by a polynomial in the size of the largest input instance, this concludes the cross-composition proof and establishes the claim. ◀

▶ **Corollary 12.** *If $\mathcal{F}$ is a class of graphs containing all cliques, then* Vertex Cover *and* Independent Set *parameterized by the minimum number of vertex deletions to obtain a graph in $\mathcal{F}$ do not admit polynomial kernels unless NP $\subseteq$ coNP/poly. In particular,* Vertex Cover *and* Independent Set *parameterized by co-cluster deletion distance or cluster deletion distance do not admit polynomial kernels unless NP $\subseteq$ coNP/poly.* ◀

4.2 Chromatic Number parameterized by Vertex Cover

In this section we give a kernelization lower bound for Chromatic Number parameterized by vertex cover, through the use of a restricted version of 3-Coloring.

▶ **Definition 13.** A graph G is a *triangle split graph* if $V(G)$ can be partitioned into sets X, Y such that $G[X]$ is an edgeless graph and $G[Y]$ is a disjoint union of vertex-disjoint triangles.

An instance of the classical problem 3-Coloring with Triangle Split Decomposition is a tuple (G, X, Y) consisting of a graph G and a partition of its vertex set into $X \cup Y$ such that $G[X]$ is edgeless and $G[Y]$ is a union of vertex-disjoint triangles. The question is whether G has a proper 3-coloring. It is not hard to show show that this restricted form of the problem is NP-complete, by replacing all edges in a normal instance of 3-Coloring with a triangle.

▶ **Theorem 14.** Chromatic Number *parameterized by the size of a vertex cover does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

Proof. To prove the theorem we will show that 3-Coloring with Triangle Split Decomposition cross-composes into Chromatic Number parameterized by a vertex cover of the graph. By a suitable choice of polynomial equivalence relation in the same style as in Theorem 11 we may assume that we are given t input instances which encode structures $(G_1, X_1, Y_1), \ldots, (G_t, X_t, Y_t)$ of 3-Coloring with Triangle Split Decomposition with $|X_i| = n$ and $|Y_i| = 3m$ for all $i \in [t]$ (i.e., m is the number of triangles in each instance). We will compose these instances into one instance (G', Z', ℓ', k') of Chromatic Number parameterized by vertex cover. By duplicating some instances we may assume that the number of inputs t is a power of 2; this only increases the input size by a factor of at most 2, and hence any bounds which are polynomial in the old input size will be polynomial in the new input size which is sufficient for our purposes.

For each set Y_i, label the triangles in $G_i[Y_i]$ as $T_1, \ldots, T_m$ in some arbitrary way, and label the vertices in each triangle T_j for a set Y_i as a_i^j, b_i^j, c_i^j. We build a graph G' with a vertex cover of size $k' := 3\log t + 4 + 3m \in O(m + \log t)$ such that G' can be $\ell' := \log t + 4$-colored if and only if one of the input instances can be 3-colored.

1. Create a clique on vertices $\{p_i \mid i \in [\log t]\} \cup \{w, x, y, z\}$; it is called the *palette*.
2. Add the vertices $\bigcup_{i=1}^{t} X_i$ to the graph, and make them adjacent to the vertex w.
3. For $i \in [m]$ add a triangle T_i^* to the graph on vertices $\{a_i, b_i, c_i\}$. The union of these triangles will be the *triangle vertices* T^*. Make all vertices in T^* adjacent to all vertices from the set $\{p_i \mid i \in [\log t]\} \cup \{w\}$.
4. For $i \in [\log t]$ add a path on two new vertices $\{q_0^i, q_1^i\}$ to the graph, and make them adjacent to all vertices $(\{p_j \mid j \in [\log t]\} \cup \{x, y, z\}) \setminus \{p_i\}$. These vertices form the *instance selector* vertices.
5. For each instance number $i \in [t]$ consider the binary representation of the value i, which can be expressed in $\log t$ bits. Consider each position $j \in [\log t]$ of this binary representation, where position 1 is most significant and $\log t$ is least significant. If bit number j of the representation of i is a 0 (resp. a 1) then make vertex q_0^j (resp. q_1^j) adjacent to all vertices of X_i. (We identify t by the all-zero string $0 \ldots 0$.)
6. As the final step we re-encode the adjacencies between vertices in the independent sets X_i and the triangles into our graph G'. For each $i \in [t]$, for each vertex $v \in Y_i$, do the following. If v is adjacent in G_i to vertex a_i^j then make vertex v adjacent in G' to a_j. Do the same for adjacencies of v to b_i^j and c_i^j.

This concludes the construction. The following claims about G' are easy to verify:

(I) In every proper $\ell' = \log t + 4$-coloring of G', the following must hold:
   - **a.** each of the $\log t + 4$ vertices of the palette clique receives a unique color,
   - **b.** consider some $i \in [\log t]$: the vertices q_0^i and q_1^i receive different colors (since they are adjacent), one of them must take the color of w and the other of p_i (they are adjacent to all other vertices of the palette),
   - **c.** the triangle vertices T^* are colored using the colors of x, y, z (they are adjacent to all other vertices of the palette),
   - **d.** the only colors which can occur on a vertex in X_i (for all $i \in [t]$) are the colors given to x, y, z and $\{p_j \mid j \in [\log t]\}$ (since the vertices in X_i are adjacent to w).

(II) For every $i \in [t]$, the graph $G'[X_i \cup T^*]$ is isomorphic to G_i.

(III) The set $Z' := \{p_i \mid i \in [\log t]\} \cup \{w, x, y, z\} \cup T^* \cup \{q_0^i, q_1^i \mid i \in [\log t]\}$ forms a vertex cover of G' of size $k' = |Z'| = 3\log t + 4 + 3m$. Hence we establish that G' has a vertex cover of size $O(m + \log t)$.

Using the given properties of G' one may verify that $\chi(G') \leq \log t + 4 \Leftrightarrow \exists i \in [t] : \chi(G_i) \leq 3$. The remainder of the proof is deferred to the full version due to space restrictions. ◀

For every fixed integer q, the q-Coloring problem parameterized by the vertex cover number *does* admit a polynomial kernel. This fact was independently observed by one of the referees. Kernelization algorithms for structural parameterizations of the q-Coloring problem will be the topic of a future publication.

4.3 Kernelization lower bounds for Feedback Vertex Set

In this section we give several kernelization lower bounds for Feedback Vertex Set. The proofs can be found in the full version [5].

▶ **Theorem 15.** *Unless NP ⊆ coNP/poly,* Feedback Vertex Set *does not admit a polynomial kernel when parameterized by 1) deletion distance to cluster graphs, and 2) deletion distance to co-cluster graphs.* ◀

▶ **Theorem 16.** Weighted Feedback Vertex Set*, where each vertex is given a positive integer as its weight, does not admit a polynomial kernel parameterized by the size of a vertex cover unless NP ⊆ coNP/poly.* ◀

5 Conclusions

We have introduced the technique of cross-composition and used it to derive kernelization lower bounds for structural parameterizations of several graph problems. Since we expect that cross-composition will be a fruitful tool in the further study of kernelization lower bounds, we give some pointers on how to devise cross-composition constructions. As the source problem of the composition one may choose a restricted yet NP-hard version of the target problem; this brings down the richness of the instances that need to be composed. If the goal is to give a lower bound for a structural parameterization (such as the size of a vertex cover) then starting from a problem on graphs which decompose into an independent set and some very structured remainder (e.g. triangle split graphs decompose into an independent set and vertex-disjoint triangles) it may be possible to compose the instances by taking the disjoint union of the inputs, and one-by-one identifying the vertices in the structured remainder. The fact that cross-compositions allow the output parameter to be polynomial in the size of the largest input can also be exploited, e.g., the proof of Theorem 11 uses this when composing input instances on n vertices into a graph G': we create $n^{O(1)}$ vertices inside a vertex cover Z' for G', and the adjacencies between Z' and a single vertex outside the cover represent the entire adjacency structure of an input graph.

Cross-composition is also appealing from a methodological point of view, since it gives a unified way of interpreting the two earlier techniques for proving kernelization lower bounds: or-compositions and polynomial-parameter transformations can both be seen to yield cross-compositions for a problem. For or-composition this is trivial to see since an or-composition for problem Q just shows that the unparameterized variant $\tilde{Q}$ cross-composes into Q. The combination of an or-composition for problem P and a polynomial-parameter transform from P to Q also gives a cross-composition: first applying the or-composition on instances of P and then transforming the resulting P-instance to a Q-instance effectively shows that we can cross-compose instances of the unparameterized variant $\tilde{P}$ into instances of Q. Hence the cross-composition technique puts the existing methods of showing super-polynomial kernelization lower bounds in a common framework, and also explains *why* these problems do not admit polynomial kernels: a parameterized problem P does not admit a polynomial kernel if it can encode the or of *some* NP-hard problem for a sufficiently small parameter value. This new perspective might lead to a deeper insight into the common structure of FPT problems without polynomial kernels.

Acknowledgements We would like to thank Holger Dell for insightful discussions which led to a more elegant proof of Theorem 9.

References

1 Peter Allen, Vadim Lozin, and Michaël Rao. Clique-width and the speed of hereditary properties. *The Electronic Journal of Combinatorics*, 16(1), 2009.

2 Hans L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proc. 4th IWPEC*, pages 17–37, 2009.
3 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.
4 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. In *Proc. 50th FOCS*, pages 629–638, 2009.
5 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Cross-composition: A new technique for kernelization lower bounds. *CoRR*, abs/1011.4224, 2010.
6 Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008.
7 Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *Proc. 17th ESA*, pages 635–646, 2009.
8 Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005.
9 Leizhen Cai. Parameterized complexity of vertex colouring. *Discrete Applied Mathematics*, 127(3):415–429, 2003.
10 Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set new measure and new structures. In *Proc. 12th SWAT*, pages 93–104, 2010.
11 Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.
12 Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *Proc. 42nd STOC*, pages 251–260, 2010.
13 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and IDs. In *Proc. 36th ICALP*, pages 378–389, 2009.
14 Rod Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, New York, 1999.
15 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*, Dagstuhl, Germany, 2009.
16 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.
17 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.
18 Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
19 Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.
20 Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving max-2-sat above a tight lower bound. *CoRR*, abs/0907.4573, 2009.
21 Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. All ternary permutation constraint satisfaction problems parameterized above average have kernels with quadratic numbers of variables. In *Proc. 18th ESA*, pages 326–337, 2010.
22 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
23 Stéphan Thomassé. A quadratic kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.
24 Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.*, 26:287–300, 1983.