




# Crossing Paths with Hans Bodlaender: A Personal View on Cross-Composition for Sparsification Lower Bounds

Bart M. P. Jansen<sup>(✉)</sup> 

Eindhoven University of Technology, Eindhoven, The Netherlands  
b.m.p.jansen@tue.nl

**Abstract.** On the occasion of Hans Bodlaender's 60th birthday, I give a personal account of our history and work together on the technique of cross-composition for kernelization lower bounds. I present several simple new proofs for polynomial kernelization lower bounds using cross-composition, interlaced with personal anecdotes about my time as Hans' PhD student at Utrecht University. Concretely, I will prove that VERTEX COVER, FEEDBACK VERTEX SET, and the  $H$ -Factor problem for every graph  $H$  that has a connected component of at least three vertices, do not admit kernels of  $\mathcal{O}(n^{2-\varepsilon})$  bits when parameterized by the number of vertices  $n$  for any  $\varepsilon > 0$ , unless  $\text{NP} \subseteq \text{coNP}/\text{poly}$ . These lower bounds are obtained by elementary gadget constructions, in particular avoiding the use of the Packing Lemma by Dell and van Melkebeek.



**Keywords:** Cross-composition · Kernelization lower bounds · Graph problems

## 1 Getting Acquainted

### 1.1 Our Meeting

Hans Bodlaender saved me from becoming a computer-game programmer.<sup>1</sup> After having spent my high-school years programming a Star Wars-themed shooter, I decided to enroll in the Computer Science program at Utrecht University. My main motivation at the time: it was the only university in the Netherlands to offer a master's degree in Game Design.

<sup>1</sup> For this special occasion, I will allow myself to write in first person.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 803421, ReduceSearch).

The original version of this chapter was revised: this chapter was previously published non-open access. The correction to this chapter is available at [https://doi.org/10.1007/978-3-030-42071-0\\_19](https://doi.org/10.1007/978-3-030-42071-0_19)

My undergraduate years passed by, during which I had my first encounters with Hans during his Algorithms course in 2005. I recall a lecture on dynamic programming, which dealt with the PRETTY PRINTING problem of dividing words over lines to minimize the sum of the squares of the unused spaces on each line. Hans mentioned that a greedy strategy does not work for this problem; you really need to use dynamic programming. A fellow student asked if one can at least use a greedy strategy to determine what the optimal number of lines is: can it be that an optimal layout uses more lines than the greedy minimum? Rather than immediately giving the answer, Hans gave the question back to us students. It fascinated me, and the remainder of the first half of the lecture passed me by as I worked out an example showing the answer to be *yes*. During the break, I showed my construction to Hans. I watched proudly as he shared my example with the rest of the class after the break. This was my first personal interaction with Hans, and a sample of how he piqued my interest in algorithmic questions. But at that time, in my second year of study, I had no inkling as to how much he would later affect my career.

The real epiphany came two years later, when I was already enrolled in the master's program on Game Design. I was following Hans' course *Algorithms and Networks*, which covered some basic concepts of parameterized complexity, when it hit me: I like programming computer games because you have to be very *efficient with your computations*: otherwise your game is going to be either slow or ugly, and in either case people will not enjoy it. The *game* aspect was ultimately not so appealing to me. So I decided to change course dramatically.

All the gaming courses I had followed up to that point were moved into the elective space of my new master's program *Applied Computing Science*. I asked Hans to supervise my master's thesis project on kernelization, to which he quickly agreed. He lent me a copy of Downey and Fellows' first textbook [18] on parameterized complexity to read up on the relevant background. I remember a distinct feeling of shock when seeing the title of Section 6.3: "Bodlaender's Theorem" in the table of contents. Up to that point, I had no idea that the friendly and humble algorithms professor at my home university was the internationally recognized authority on treewidth!

Several months into my final project, Hans told me that he was writing a grant proposal to acquire funding to study kernelization, and asked me if I would mind being mentioned in the proposal as a qualified candidate for the PhD position he was requesting. A couple of months later, nearing the end of my master thesis project, I had proven my first kernelization results when Hans told me the good news: his NWO TOP grant "KERNELS: Combinatorial Analysis of Data Reduction" was funded. That led to my easiest job interview ever, which consisted of two lines. Hans: "So, do you still want to become a PhD student?" to which my answer was a resounding "Yes"! I never looked back.

## 1.2 Our Work on Kernelization Lower Bounds

I started working on my PhD under Hans' supervision in 2009, investigating the power and limitations of efficient and provably effective preprocessing. When

Stefan Kratsch joined the research group in 2010, we were completed into a trio that studied kernelization during the day and played boardgames at night.<sup>2</sup> Kernelization theorems were discussed at *Chez Hans*, the nickname that Hans' office earned for the clandestine coffee machine that served many of our colleagues.<sup>3</sup>

Building on a series of papers that had just come out [3, 4, 22, 23], we spent a lot of time working on hardness proofs showing that small kernels do not exist under certain complexity-theoretic assumptions. For this contribution to the festschrift on account of Hans' 60th birthday, I therefore decided to write about kernelization lower bounds based on the framework of cross-composition [5, 7] developed by Hans, Stefan, and myself. The purpose of the technical content of this article is to show a number of new and elegant kernelization lower-bound proofs based on cross-composition, showing that VERTEX COVER,  $H$ -FACTOR, and FEEDBACK VERTEX SET parameterized by the number of vertices  $n$  cannot be efficiently reduced to equivalent instances on  $\mathcal{O}(n^{2-\varepsilon})$  bits for any  $\varepsilon > 0$ , unless  $\text{NP} \subseteq \text{coNP}/\text{poly}$ . The first two yield new proofs for existing theorems, but the third result is new. The proofs are elementary, based on the cross-composition framework with simple gadgeteering. All lower bounds will follow from combining a suitable choice of starting problem for the reduction, together with gadgets to deactivate most parts of the composed instance. In particular, all these lower bounds can be proven without having to resort to the Packing Lemma of Dell and van Melkebeek [16].

### 1.3 Organization

In Sect. 2 I introduce the prerequisite definitions of parameterized complexity and kernelization, together with the framework of cross-composition. The remainder of the article is organized into three case studies. Each case study deals with one problem, discussing the relevant related work before presenting a kernelization lower bound using cross-composition. Section 3 deals with parameterized complexity's fruit fly, VERTEX COVER. More general vertex-deletion problems such as FEEDBACK VERTEX SET are considered in Sect. 4. Section 5 focuses on the  $H$ -FACTOR problem. Finally, I give some concluding reflections in Sect. 6.

## 2 Kernelization and Lower Bounds

### 2.1 Kernelization

Kernelization investigates how a complexity *parameter* contributes to the difficulty of an input to an algorithmic decision problem, and therefore uses notions

<sup>2</sup> I have particularly fond memories of Robo Rally, or *applied algorithmic game theory* as Hans liked to put it, in which you plan a sequence of actions for your robot in the hope of it being the first to visit all checkpoints. More often than not, it ends up pushed into a pit or shot at by competing robots which simultaneously carry out the instructions that their governing players programmed.

<sup>3</sup> Hans did not drink coffee until his fifties, strategically saving the caffeine-filled theorem-producing beverage until it was needed. I am following in his footsteps and have never had a single cup of coffee.

from parameterized complexity [13, 19–21]. Fix a finite alphabet  $\Sigma$  used to encode problem inputs, such as  $\Sigma = \{0, 1\}$ . A *parameterized problem* is a subset  $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ , which contains the pairs  $(x, k)$  for which  $x$  is the encoding of an input to  $\mathcal{Q}$  whose parameter value is  $k$ , and for which the answer to the encoded question is YES. Intuitively, a *kernelization* for a parameterized problem  $\mathcal{Q}$  is a polynomial-time preprocessing algorithm that reduces any input  $(x, k)$  to an equivalent one, whose size and parameter are bounded solely in terms of the complexity parameter  $k$ , independent of the size  $|x|$  of the original input. The lower bounds I present turn out to work against the more general notion of *generalized kernelization*, which essentially investigates whether inputs of one parameterized problem  $\mathcal{Q}$  can be efficiently reduced to small instances of another problem  $\mathcal{Q}'$ . I therefore need the following formal definition.

**Definition 1.** *Let  $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$  be parameterized problems and let  $h: \mathbb{N} \rightarrow \mathbb{N}$  be a computable function. A generalized kernel for  $\mathcal{Q}$  into  $\mathcal{Q}'$  of size  $h(k)$  is an algorithm that, on input  $(x, k) \in \Sigma^* \times \mathbb{N}$ , takes time polynomial in  $|x| + k$  and outputs an instance  $(x', k')$  such that:*

1.  $|x'|$  and  $k'$  are bounded by  $h(k)$ , and
2.  $(x', k') \in \mathcal{Q}'$  if and only if  $(x, k) \in \mathcal{Q}$ .

*The algorithm is a kernel for  $\mathcal{Q}$  if  $\mathcal{Q}' = \mathcal{Q}$ . It is a polynomial (generalized) kernel if  $h(k)$  is a polynomial.*

Much of the initial research on kernelization lower bounds [4, 6, 8, 17] dealt with the distinction between polynomial and super-polynomial kernel sizes. Later, a refinement of the tools made it possible to also give lower bounds on the degree of the polynomial that bounds the kernel size, for problems that *do* admit a polynomial-size kernel. In this article, I will focus on the latter kind of lower bounds. For a historical overview of the development of kernelization lower bounds, I refer to the introductory sections of the article on cross-composition by myself, Hans Bodlaender, and Stefan Kratsch [7].

## 2.2 Intuition for Polynomial Kernelization Lower Bounds

To show the impossibility (under suitable complexity-theoretic conjectures) of a parameterized problem  $\mathcal{Q}$  having small kernels, one must prove that the existence of a small kernel would imply algorithmic consequences that are “too good to be true” and therefore violate established beliefs such as  $\text{P} \neq \text{NP}$ . Before presenting the formal details, let me try to convey some intuition behind such proofs, elaborating on what these consequences are. Consider the NP-hard VERTEX COVER problem, whose inputs consist of a graph  $G$  and integer  $k$ . The question is whether  $G$  has a set  $S$  of at most  $k$  vertices, such that each edge of  $G$  contains at least one vertex from  $S$ . Using the Nemhauser-Trotter theorem [31], an instance  $(G, k)$  can efficiently be reduced to an equivalent instance  $(G', k')$  where  $G'$  has at most  $2k$  vertices and  $k' \leq k$ . The kernelized instance can be encoded in  $\mathcal{O}(k^2)$  bits, by writing down the adjacency matrix of  $G'$  and the

integer  $k'$  in unary. How could one prove that it is impossible to always obtain a kernel of, say,  $\mathcal{O}(k^{1.9})$  bits?

Let us consider algorithmic consequences which are “too good to be true”. First of all, suppose there would be a polynomial-time algorithm  $\mathcal{A}$  that, given a sequence of  $t$  instances  $(G_1, k_1), \dots, (G_t, k_t)$  of VERTEX COVER, would be able to distinguish between the case that all input instances have answer NO, and the case that there is at least one YES-instance among the inputs. Then using  $\mathcal{A}$ , we could solve the VERTEX COVER decision problem in polynomial time: to determine the answer to  $(G, k)$ , just ask  $\mathcal{A}$  whether a sequence consisting of some arbitrarily chosen NO-instances, together with the instance  $(G, k)$ , contains at least one YES-instance. Hence such an algorithm  $\mathcal{A}$  is too good to be true.

Now suppose that there is an algorithm  $\mathcal{B}$  that, given a sequence of  $t \in \mathcal{N}^{\mathcal{O}(1)}$  instances of VERTEX COVER, each on  $N$  bits, outputs a single VERTEX COVER instance  $(G^*, k^*)$  on  $(t - 1) \cdot N$  bits whose answer is YES if and only if there was a YES-instance in the input sequence. Even though such an algorithm  $\mathcal{B}$  does not directly give a way to solve VERTEX COVER in polynomial time, I will argue it is still too good to be true. To achieve an output size of  $(t - 1) \cdot N$  bits, intuitively it has to omit one of the input instances when building the output  $(G^*, k^*)$ . But to ensure that the output  $(G^*, k^*)$  accurately represents the logical OR of the input sequence, it seems that  $\mathcal{B}$  must solve an input instance before it can be sure that it is safe to omit it. After all, if the omitted instance was the only one in the sequence whose answer was YES, then omitting it changes the value of the logical OR. Since we believe VERTEX COVER cannot be solved in polynomial time, it becomes intuitively clear that algorithm  $\mathcal{B}$  is also too good to be true.

We can therefore prove the impossibility of having kernels of bitsize  $\mathcal{O}(k^{1.9})$  by developing a so-called cross-composition algorithm  $\mathcal{C}$  which, together with such a kernel, would yield algorithm  $\mathcal{B}$ . Suppose  $\mathcal{C}$  can take any sequence of  $t$  instances of VERTEX COVER, each on  $n$  vertices and therefore  $N \in \mathcal{O}(n^2)$  bits, and composes these into a single instance  $(G', k')$  whose answer is the logical OR of the answers to the inputs, on  $\mathcal{O}(\sqrt{t} \cdot n)$  vertices such that  $k' \in \mathcal{O}(\sqrt{t} \cdot n)$ . Suppose further that  $\mathcal{D}$  is a kernelization for VERTEX COVER of bitsize  $\mathcal{O}(k^{1.9})$ . Then by pipe-lining algorithms  $\mathcal{C}$  and  $\mathcal{D}$ , we obtain an algorithm such as  $\mathcal{B}$ , thereby showing that  $\mathcal{D}$  should not exist. Indeed, if we take a sequence of  $t = n^{50}$  instances of VERTEX COVER, each on  $n$  vertices, then the composition  $\mathcal{C}$  merges these into a single instance with parameter value  $k' \in \mathcal{O}(\sqrt{t} \cdot n) = \mathcal{O}(n^{26})$ . Applying the kernelization  $\mathcal{D}$  to this composed instance, reduces it to size  $\mathcal{O}((n^{26})^{1.9}) \leq \mathcal{O}(n^{49.4}) < (t - 1) \cdot N \approx n^{52}$ , therefore forming an algorithm of type  $\mathcal{B}$  that is too good to be true; hence if we can find such a cross-composition  $\mathcal{C}$ , then such a kernel  $\mathcal{D}$  should not exist. By increasing the number of input instances from  $n^{50}$  to larger powers of  $n$ , the same intuitive reasoning rules out the existence of a kernel of bitsize  $\mathcal{O}(k^{2-\varepsilon})$  for any  $\varepsilon > 0$ .

Note that the key property of  $\mathcal{C}$  that makes this work, is that it manages to compress the information of  $t$  instances on  $n$  vertices each, into a single instance on  $\mathcal{O}(\sqrt{t} \cdot n)$  vertices. While this may seem far-fetched at first, there are elementary reductions achieving this. They exploit the fact that  $t$  instances on  $n$

vertices carry  $t \cdot n^2$  bits of information (for each of the  $t$  graphs, which of the  $n^2$  potential edges exist?) while a single instance on  $\mathcal{O}(\sqrt{t} \cdot n)$  vertices has  $\mathcal{O}(t \cdot n^2)$  potential edges, and can therefore encode the same amount of information if it is packed efficiently. Now let me make this intuition precise.

### 2.3 The Formal Cross-Composition Framework

The fact that algorithms such as  $\mathcal{B}$  are too good to be true<sup>4</sup> was proven by Dell and van Melkebeek [16, §6], building on work by Fortnow and Santhanam [23], which in turn was triggered by the seminal work on kernelization lower bounds by Hans with Downey, Fellows, and Hermelin [4]. The existence of algorithms like  $\mathcal{B}$  does not directly lead to the consequence that  $\text{P} = \text{NP}$ , but implies the complexity-theoretic containment  $\text{NP} \subseteq \text{coNP}/\text{poly}$ , which is still considered very unlikely. To prove kernelization lower bounds, we therefore formalize what the composition algorithm, referred to above as  $\mathcal{C}$ , has to achieve.

In many cases, it turns out to be easier to build composition algorithms for sequences of “similarly-sized” input instances  $(G_1, k_1), \dots, (G_t, k_t)$ , for example when the input graphs all have the same number of vertices, edges, and target vertex cover size  $k$ . Since there are exponentially many different instances of a given number of vertices, edges, and target cover size, such a restriction does not make the algorithmic task much easier. Indeed, even if the composition algorithm  $\mathcal{C}$  described above is only applied to sequences of similarly-sized inputs, the combination of  $\mathcal{C}$  and  $\mathcal{D}$  still leads to algorithms that are too good to be true. For that reason, the cross-composition framework [7] allows one to choose an equivalence relation on inputs, efficiently grouping inputs of bitsize  $N$  into  $N^{\mathcal{O}(1)}$  different classes, and only requires a composition algorithm to be able to merge inputs coming from the same class.

**Definition 2 (Polynomial equivalence relation).** *An equivalence relation  $\mathcal{R}$  on  $\Sigma^*$  is called a polynomial equivalence relation if the following 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 time polynomial in  $|x| + |y|$ .*
2. *For any finite set  $S \subseteq \Sigma^*$  the equivalence relation  $\mathcal{R}$  partitions the elements of  $S$  into a number of classes that is polynomially bounded in the size of the largest element of  $S$ .*

Definition 2 allows one to circumvent padding arguments that were frequently used in earlier proofs. Using this notion, we can now formalize cross-composition for proving polynomial lower bounds on kernelization. For this overview article, rather than defining bounded-cost cross-composition in general (cf. [7, §3.2]), I restrict myself to the less technical degree-2 cross-compositions like the ones described above, which are used to rule out kernels of subquadratic size. In the following definition, I use the shorthand  $[n] = \{1, \dots, n\}$  for  $n \in \mathbb{N}$ .

<sup>4</sup> The formal details differ slightly from my intuitive interpretation above.

**Definition 3 (Degree-2 cross-composition).** *Let  $L \subseteq \Sigma^*$  be a language, let  $\mathcal{R}$  be a polynomial equivalence relation on  $\Sigma^*$ , and let  $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$  be a parameterized problem. A degree-2 OR-cross-composition of  $L$  into  $\mathcal{Q}$  with respect to  $\mathcal{R}$  is an algorithm that, given  $t$  instances  $x_1, x_2, \dots, x_t \in \Sigma^*$  of  $L$  belonging to the same equivalence class of  $\mathcal{R}$ , takes time polynomial in  $\sum_{i=1}^t |x_i|$  and outputs an instance  $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$  such that:*

1. *the parameter  $k^*$  is bounded by  $\mathcal{O}(\sqrt{t} \cdot (\max_i |x_i|)^c)$ , where  $c$  is some constant independent of  $t$ , and*
2.  *$(x^*, k^*) \in \mathcal{Q}$  if and only if there is an  $i \in [t]$  such that  $x_i \in L$ .*

The adjective *cross* in the name cross-composition comes from the fact that the reduction crosses over from inputs of problem  $L$ , into an input of parameterized problem  $\mathcal{Q}$ . This contrasts the earlier plain *composition* framework of Bodlaender, Downey, Fellows, and Hermelin [4] which required problems to be composed into themselves. As we will see, crossing over from one problem to another makes it easier to prove lower bounds in several settings.

When building a degree-2 OR-cross-composition, it will be convenient if the number of input instances  $t$  is a square: then  $\sqrt{t} \in \mathbb{N}$ , which allows the input instances to be enumerated as  $x_{i,j}$  for  $i, j \in [\sqrt{t}]$ . This assumption can be made without loss of generality. Suppose we have a cross-composition  $\mathcal{C}$  that works if  $t$  is a square, and we want to obtain a cross-composition  $\mathcal{C}'$  for an arbitrary number of inputs. Then  $\mathcal{C}'$  can be obtained as follows. Given an input sequence  $x_1, \dots, x_t$  to  $\mathcal{C}'$ , let  $t' \geq t$  be the smallest square that is larger than  $t$ . Note that  $t' \leq 2t$  because the nearest power of two suffices. Then build a new sequence of  $t'$  inputs by appending  $t' - t$  copies of  $x_1$  to  $x_1, \dots, x_t$ , and apply  $\mathcal{C}$  to this sequence. Clearly the logical OR of the old and new sequences have the same value, and it is easy to verify that  $\mathcal{C}'$  satisfies all conditions of Definition 3. We will therefore assume without loss of generality that  $t$  is a square.

The following theorem shows that degree-2 OR-cross-compositions indeed rule out subquadratic kernels, under the assumption that  $\text{NP} \not\subseteq \text{coNP/poly}$ .

**Theorem 1** ([7, Thm. 3.8, Prop. 2.3]). *Let  $L \subseteq \Sigma^*$  be a language that is NP-hard under Karp reductions, let  $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$  be a parameterized problem, and let  $\varepsilon > 0$  be a real number. If  $L$  is NP-hard under Karp reductions and has a degree-2 OR-cross-composition into  $\mathcal{Q}$ , and  $\mathcal{Q}$  parameterized by  $k$  has a polynomial (generalized) kernelization of bitsize  $\mathcal{O}(k^{2-\varepsilon})$ , then  $\text{NP} \subseteq \text{coNP/poly}$ .*

In the upcoming case studies, I will consider a number of graph problems parameterized by the number of vertices  $n$ . Hence the parameter value, denoted by  $k$  in the definitions of the framework, will be the number of vertices of the input graph that is commonly denoted as  $n$ . Any pair  $(G, k)$  consisting of an  $n$ -vertex graph, together with a target value  $k$  in the range  $\{0, \dots, n\}$ , can trivially be encoded in  $\mathcal{O}(n^2)$  bits. Graph problems whose input is of the form  $(G, k)$  therefore have trivial polynomial kernels of bitsize  $\mathcal{O}(n^2)$  when parameterized by the number of vertices  $n$ . The lower bounds will prove that this cannot be significantly improved. So intuitively, the lower bounds will rule out that there

is an efficient *sparsification* algorithm that reduces a dense  $n$ -vertex instance to an equivalent one with a subquadratic number of edges. Note that such a sparsification bound directly implies that, for any problem parameter  $\ell$  whose value on  $n$ -vertex graphs is  $\mathcal{O}(n)$ , there cannot be a kernel of bitsize  $\mathcal{O}(\ell^{2-\varepsilon})$ .

### 3 Vertex Cover

The first case study concerns the VERTEX COVER problem. The problem admits a simple degree-based kernelization due to Sam Buss [11] that reduces inputs  $(G, k)$  to  $\mathcal{O}(k^2)$  vertices and edges. The same bounds can be obtained via the sunflower lemma [20, §9.1]. The linear-programming based kernelization based on the Nemhauser-Trotter theorem [31] achieves a better bound of  $2k$  vertices, but may still have  $\Omega(k^2)$  edges. In a breakthrough paper [15, 16], Dell and van Melkebeek proved that this is optimal up to  $k^{o(1)}$  factors: they proved that VERTEX COVER has no kernel of bitsize  $\mathcal{O}(k^{2-\varepsilon})$  for any  $\varepsilon > 0$ , unless  $\text{NP} \subseteq \text{coNP/poly}$ . Their proof is based on a nontrivial number-theoretic construction called the *Packing Lemma* [16, Lemma 2], which shows how to construct graphs whose edges partition into many large cliques, in such a way that no large cliques exist other than in the packing. Later, Dell and Marx [14, Thm. C.1] showed how to obtain the same lower bound using an elementary gadget construction, by composing instances of MULTICOLORED BICLIQUE based on a table layout (cf. [21, §20.2]).

To illustrate the technique of degree-2 OR-cross-composition, I will present an alternative elementary lower-bound construction for VERTEX COVER, which avoids the intermediate problem of MULTICOLORED BICLIQUE. It will be useful to use a restricted version of VERTEX COVER as a starting point for the composition, though, which is formalized in the following way.

#### VERTEX COVER ON SUBDIVIDED GRAPHS

**Input:** A graph  $G$ , an integer  $k$ , and a partition of  $V(G)$  into  $A \cup B$  such that  $G[A]$  is an independent set and each connected component of  $G[B]$  consists of a single edge.

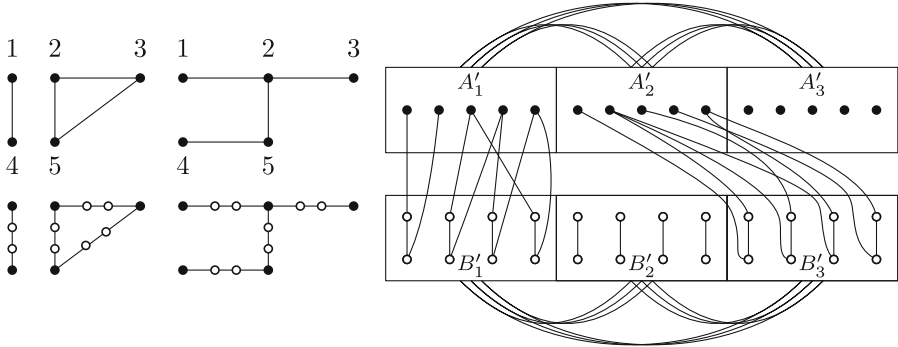
**Question:** Does  $G$  contain a set  $S \subseteq V(G)$  of size at most  $k$ , such that  $S \cap \{u, v\} \neq \emptyset$  for each edge  $\{u, v\} \in E(G)$ ?

The partition of  $V(G)$  into sets  $A$  and  $B$  that induce subgraphs of a specific form will be useful when merging a series of inputs into one. Intuitively, it will allow us to merge the sets  $A$  of various distinct inputs into a single set, while preserving the adjacency information of the original inputs.

**Lemma 1.** VERTEX COVER ON SUBDIVIDED GRAPHS is NP-hard under Karp reductions.

*Proof.* Consider an instance  $(G, k)$  of VERTEX COVER, and pick an arbitrary edge  $\{x, y\}$ . Let  $G'$  be the graph obtained from  $G$  by removing the edge  $\{x, y\}$ ,





**Fig. 1.** Left: A 5-vertex graph at the top; below it, the graph obtained by subdividing every edge twice, which is used as input  $G_{1,1}$  to the cross-composition. In the subdivided instance, the set  $A$  of original vertices is in black, the set  $B$  of subdividers is in white. Middle: Another 5-vertex graph with its double-subdivision, used as instance  $G_{2,3}$ . Right: Illustration of the cross-composition of Theorem 2 for  $t = 3 \cdot 3$  inputs of the subdivided problem with 5 vertices in  $A$  and 8 vertices in  $B$ . Edges between different sets  $A'_i, A'_{i'}$  are visualized schematically, similarly for  $B'_j, B'_{j'}$ . Only the edges inserted on account of instances  $G_{1,1}$  and  $G_{2,3}$  are shown.

inserting two new vertices  $x', y'$  and the edges  $\{x, x'\}, \{x', y'\}, \{y', y\}$ . Intuitively,  $G'$  is obtained by subdividing the edge  $\{x, y\}$  twice. It is easy to verify that  $G$  has a vertex cover of size  $k$  if and only if  $G'$  has a vertex cover of size  $k + 1$ , which was first observed by Poljak [32] for the complementary problem INDEPENDENT SET (cf. [9, Lemma 7]).

To prove the lemma, we use the following reduction from the NP-complete VERTEX COVER problem. Given an instance  $(G, k)$  on  $m = |E(G)|$  edges, let  $G'$  be obtained by replacing each edge of  $G$  by a three-edge path as above, and let  $k' := k + m$ . By the observation above,  $G$  has a vertex cover of size  $k$  if and only if  $G'$  has a vertex cover of size  $k'$ . Letting  $A = V(G)$  denote the original vertices in  $G'$ , and letting  $B$  denote the inserted subdivider vertices, we have that  $G'[A]$  is an independent set and  $G'[B]$  consists of isolated edges. Hence  $(G', k', A, B)$  is a valid equivalent instance of VERTEX COVER ON SUBDIVIDED GRAPHS.  $\square$

Using this starting problem, I now present the degree-2 OR-cross-composition that rules out subquadratic kernels for VERTEX COVER. Refer to Fig. 1 for an illustration.

**Theorem 2.** *For any  $\varepsilon > 0$ , VERTEX COVER parameterized by the number of vertices  $n$  does not admit a generalized kernelization of bitsize  $\mathcal{O}(n^{2-\varepsilon})$  unless  $\text{NP} \subseteq \text{coNP/poly}$ .*

*Proof.* By Theorem 1 and Lemma 1 it suffices to give a cross-composition of VERTEX COVER ON SUBDIVIDED GRAPHS into VERTEX COVER, such that any sequence of  $t$  inputs of bitsize at most  $N$  each, is composed into a single instance  $(G', k')$  on  $n \in \mathcal{O}(\sqrt{t} \cdot N^{\mathcal{O}(1)})$  vertices.

Assume without loss of generality that  $t$  is a square. We define a polynomial equivalence relation  $\mathcal{R}$  so that two well-formed instances are equivalent if they agree on the number of vertices in  $A$ , the number of vertices in  $B$ , and on the target value  $k$ . Enumerate the inputs as  $(G_{i,j}, k, A_{i,j}, B_{i,j})$  for  $i, j \in [\sqrt{t}]$ , such that all inputs have  $|A_{i,j}| = n_A$  and  $|B_{i,j}| = n_B$ . Build a graph  $G'$  as follows.

1. For each  $i \in [\sqrt{t}]$ , add a vertex set  $A'_i$  of  $n_A$  independent vertices to  $G'$ .
2. For each  $i \in [\sqrt{t}]$ , add a vertex set  $B'_i$  of  $n_B$  vertices to  $G'$ . Insert edges to ensure  $G'[B'_i]$  consists of  $n_B/2$  isolated edges.
3. For each  $i \neq i' \in [\sqrt{t}]$ , add all possible edges between  $A'_i$  and  $A'_{i'}$ .
4. For each  $j \neq j' \in [\sqrt{t}]$ , add all possible edges between  $B'_j$  and  $B'_{j'}$ .
5. For each  $i, j \in [\sqrt{t}]$ , insert edges between  $A'_i$  and  $B'_j$  so that  $G'[A'_i \cup B'_j]$  is isomorphic to  $G_{i,j}$ .

To complete the cross-composition, we set  $k' := (\sqrt{t} - 1)(n_A + n_B) + k$ . Observe that  $G'$  has  $\sqrt{t} \cdot (n_A + n_B)$  vertices, which is suitably bounded for a degree-2 OR-cross-composition since input instances have  $N \geq n_A + n_B$  bits. The construction can easily be performed in polynomial time. It remains to verify that  $G'$  has a vertex cover of size  $k'$  if and only if some input instance  $G_{i,j}$  has a vertex cover of size  $k$ .

Suppose first that there is a YES-instance  $G_{i^*,j^*}$  among the inputs that has a vertex cover of size at most  $k$ . Since  $G'[A'_{i^*}, B'_{j^*}]$  is isomorphic to  $G_{i^*,j^*}$  by construction, it has a vertex cover  $S'_{i^*,j^*}$  of size at most  $k$ . Combined with all  $(\sqrt{t} - 1)(n_A + n_B)$  remaining vertices of  $G'$ , this yields a vertex cover of size at most  $k'$  of  $G'$ , proving that the result of the composition is a YES-instance.

For the other direction, suppose  $S'$  is a vertex cover of size at most  $k'$  in  $G'$ . Since all possible edges are present between distinct sets  $A'_i$  and  $A'_{i'}$ , there is at most one set  $A'_{i^*}$  from which  $S'$  does not contain all vertices. Similarly, there is at most one set  $B'_{j^*}$  from which  $S'$  does not contain all vertices. Since  $|S'| \leq k' = (\sqrt{t} - 1)(n_A + n_B) + k$ , while  $S'$  contains all  $(\sqrt{t} - 1)(n_A + n_B)$  vertices of  $(\bigcup_{i \neq i^*} A'_i) \cup (\bigcup_{j \neq j^*} B'_j)$ , it follows that  $S' \cap (A'_{i^*} \cup B'_{j^*}) \leq k$ . Since  $G'[A'_{i^*} \cup B'_{j^*}]$  is isomorphic to  $G_{i^*,j^*}$ , this proves that  $G_{i^*,j^*}$  has a vertex cover of size at most  $k$ , so that there is a YES-instance among the inputs.  $\square$

Let me point out two crucial features of the cross-composition above. First, note that in Step 5 of the construction we heavily exploit the fact that all graphs  $G_{i,j}[A_{i,j}]$  are isomorphic, and similarly that all graphs  $G_{i,j}[B_{i,j}]$  are isomorphic. The fact that the vertex sets of the input graphs can be partitioned into two parts that induce very uniformly structured subgraphs, will also be exploited in the upcoming lower bounds. Second, we used some problem-specific gadgeteering to ensure that solutions to  $G'$  must contain all but one of the groups  $A'_i$  in their entirety, and similarly for the groups  $B'_j$ . The step of inserting all possible edges between the groups ensures that effectively, a good solution in a single instance  $G_{i^*,j^*} = G'[A'_{i^*} \cup B'_{j^*}]$  is sufficient to guarantee the existence of a good solution in  $G'$ . More involved gadgeteering will be needed to achieve a similar OR behavior in future constructions.

## 4 Feedback Vertex Set

We move on to another classic vertex-deletion problem, FEEDBACK VERTEX SET. We consider the problem on *undirected* graphs; in fact, I will consider only undirected graphs throughout this article. An instance  $(G, k)$  therefore consists of an undirected graph  $G$  and integer  $k$ , and asks whether  $G$  has a subset  $S$  of at most  $k$  vertices, whose removal from  $G$  leaves an acyclic graph. Several polynomial kernels were developed for the problem [2, 10, 25, 33], one of which is famously due to Hans [1]. The current-best kernel [25] has  $\mathcal{O}(k^2)$  vertices and edges, and can be encoded in  $\mathcal{O}(k^2 \log k)$  bits. Dell and van Melkebeek [16] show that FEEDBACK VERTEX SET does not have kernels of bitsize  $\mathcal{O}(k^{2-\epsilon})$  unless  $\text{NP} \subseteq \text{coNP/poly}$ . However, their proof does not say anything about the possibility of sparsifying  $n$ -vertex instances to  $\mathcal{O}(n^{2-\epsilon})$  bits. I will show that the latter is also impossible, assuming  $\text{NP} \not\subseteq \text{coNP/poly}$ , by adapting the construction of Theorem 2. We will again need a version of the problem whose vertex set partitions into two parts that induce uniformly structured subgraphs. Since FEEDBACK VERTEX SET remains NP-complete [27] on bipartite graphs (subdividing every edge preserves the answer to the problem, and yields a bipartite graph), the following NP-complete problem will be used as the source problem for the cross-composition.

**FEEDBACK VERTEX SET ON BIPARTITE GRAPHS**

**Input:** An undirected graph  $G$ , an integer  $k$ , and a partition of  $V(G)$  into  $A \cup B$  such that  $G[A]$  and  $G[B]$  are both independent sets.

**Question:** Does  $G$  contain a vertex set  $S \subseteq V(G)$  of size at most  $k$ , such that  $G - S$  is acyclic?

**Theorem 3.** *For any  $\epsilon > 0$ , FEEDBACK VERTEX SET parameterized by the number of vertices  $n$  does not admit a generalized kernelization of bitsize  $\mathcal{O}(n^{2-\epsilon})$  unless  $\text{NP} \subseteq \text{coNP/poly}$ .*

*Proof.* I present a degree-2 OR-cross-composition. Let  $(G_{i,j}, k, A_{i,j}, B_{i,j})$  for  $i, j \in [\sqrt{t}]$  be a sequence of  $t$  input instances of FEEDBACK VERTEX SET ON BIPARTITE GRAPHS that all share the same target value  $k$ , the same number  $n_A$  of vertices in the  $A$ -set, and the same number  $n_B$  of vertices in the  $B$ -set, which may be assumed by a suitable choice of  $\mathcal{R}$ . Build an instance  $(G', k')$  as follows.

1. For each  $i \in [\sqrt{t}]$ , add a set  $A'_i$  of  $n_A$  independent vertices to  $G'$  and number these from 1 to  $n_A$ .
2. For each  $i \in [\sqrt{t}]$ , add a set  $B'_i$  of  $n_B$  independent vertices to  $G'$  and number these from 1 to  $n_B$ .
3. For each  $i, j \in [\sqrt{t}]$ , insert edges between  $A'_i$  and  $B'_j$  so that  $G'[A'_i \cup B'_j]$  is isomorphic to  $G_{i,j}$ .
4. For each  $i \in [\sqrt{t}]$ , add a vertex set  $A_i^* = \{a_{i,x,y,c}^* \mid x, y \in [n_A], c \in [2]\}$  to  $G'$ . For each  $i < i' \leq \sqrt{t}$ , for each  $x, y \in [n_A]$ , for each  $c \in [2]$ , make  $a_{i,x,y,c}^*$  adjacent to the  $x$ th vertex of  $A'_i$  and the  $y$ th vertex of  $A'_{i'}$ .

5. For each  $j \in [\sqrt{t}]$ , add a vertex set  $B_j^* = \{b_{j,x,y,c}^* \mid x, y \in [n_B], c \in [2]\}$  to  $G'$ . For each  $j < j' \leq \sqrt{t}$ , for each  $x, y \in [n_B]$ , for each  $c \in [2]$ , make  $b_{j,x,y,c}^*$  adjacent to the  $x$ th vertex of  $B_j'$  and the  $y$ th vertex of  $B_{j'}'$ .

Define  $A^* := \bigcup_i A_i^*$  and  $B^* := \bigcup_j B_j^*$ . By the last two steps, for each  $i \in [\sqrt{t}]$ , each vertex of  $A^*$  is adjacent to at most one vertex in  $A_i'$ , and symmetrically for adjacencies of  $B^*$  into sets  $B_j'$ . For every pair of vertices  $x \in A_i', y \in A_{i'}'$  for  $i < i'$ , there are two vertices  $a_{i,x,y,1}^*$  and  $a_{i,x,y,2}^*$  adjacent to both  $x$  and  $y$ . Effectively, these form a cycle with  $x$  and  $y$ , prompting the following observation.

**Observation 1.** *If  $S'$  is a feedback vertex set in  $G'$ , and  $i < i' \in [\sqrt{t}]$  such that there exist  $x \in A_i' \setminus S'$  and  $y \in A_{i'}' \setminus S'$ , then  $S'$  contains a vertex of  $\{a_{i,x,y,1}^*, a_{i,x,y,2}^*\}$ . The analogous statement for  $B'$  also holds.*

To complete the cross-composition, we set  $k' := (\sqrt{t} - 1)(n_A + n_B) + k$ . Observe that  $G'$  has  $\mathcal{O}(\sqrt{t} \cdot (n_A + n_B)^2)$  vertices, suitably bounded for a degree-2 OR-cross-composition. It remains to verify that  $G'$  has a feedback vertex set of size  $k'$  if and only if some input instance  $G_{i,j}$  has one of size  $k$ .

Suppose first that  $G_{i^*,j^*}$  has a feedback vertex set of size  $k$ . Since  $G'[A_{i^*}' \cup B_{j^*}']$  is isomorphic to  $G_{i^*,j^*}$ , it has a feedback vertex set  $S_{i^*,j^*}$  of size  $k$ . Consider  $S' := S_{i^*,j^*} \cup (\bigcup_{i \neq i^*} A_i') \cup (\bigcup_{j \neq j^*} B_j')$ , which has size at most  $k'$ . Now observe that  $G' - S'$  can be obtained from the acyclic graph  $G'[A_{i^*}' \cup B_{j^*}'] - S_{i^*,j^*}$  by inserting the vertices  $A^* \cup B^*$  along with their edges into  $(A_{i^*}' \cup B_{j^*}') \setminus S_{i^*,j^*}$ . As each vertex of  $A^*$  is adjacent to at most one vertex of  $A_{i^*}'$ , and each vertex of  $B^*$  is adjacent to at most one vertex of  $B_{j^*}'$ , the graph  $G' - S'$  is obtained from an acyclic graph by inserting vertices of degree at most one, which does not introduce any cycles. Hence  $S'$  is a feedback vertex set of size at most  $k'$  in  $G'$ .

For the reverse direction, suppose that  $G'$  has a feedback vertex set  $S'$  of size at most  $k'$ . If there are distinct indices  $i < i' \in [\sqrt{t}]$  for which  $A_i' \setminus S'$  and  $A_{i'}' \setminus S'$  are both nonempty, then normalize  $S'$  as follows. Let  $i^*$  be the largest index for which  $A_{i^*}' \setminus S' \neq \emptyset$ , and define  $S'' := (S' \setminus A^*) \cup (\bigcup_{i \neq i^*} A_i')$ . Since all vertices of  $A^*$  have at most one neighbor in  $A_{i^*}'$ , they have degree at most one in  $G' - S''$  and therefore  $G' - S''$  is also acyclic. Let me show that  $|S''| \leq |S'|$ . For each  $i < i^*$ , for each vertex  $x \in A_i' \setminus S'$ , vertex  $x$  belongs to  $S''$  but not to  $S'$ . To show  $S''$  is not larger than  $S'$ , we charge each such  $x$  to a unique vertex that is contained in  $S'$  but not in  $S''$ . Fix an arbitrary  $y \in A_{i^*}' \setminus S'$ . By Observation 1 the solution  $S'$  contains a vertex of  $\{a_{i,x,y,1}^*, a_{i,x,y,2}^*\}$  to which we can charge  $x \in A_i' \setminus S'$ . In this way we can charge each  $x \in S'' \setminus S'$  to a unique pair, implying that  $|S''| \leq |S'|$ . Hence this normalization process yields a feedback vertex set  $S^*$  of size at most  $k'$  that contains all vertices of  $\bigcup_{i \neq i^*} A_i'$  for a suitable choice of  $i^* \in [\sqrt{t}]$ . By a second analogous and independent normalization step for  $B'$ , we may assume there is an index  $j^*$  such that  $S^*$  contains all vertices of  $\bigcup_{j \neq j^*} B_j'$ . Since  $|S^*| \leq k' = (\sqrt{t} - 1)(n_A + n_B) + k$ , the set  $S^*$  contains at most  $k$  vertices from  $G'[A_{i^*}' \cup B_{j^*}']$ , which is isomorphic to  $G_{i^*,j^*}$  by construction. Hence  $G_{i^*,j^*}$  has a feedback vertex set of size at most  $k$ .  $\square$

The type of construction of Theorem 3 can be used to prove analogous lower bounds for many other vertex-deletion problems to nontrivial hereditary graph classes: all one has to do is change the source problem of the composition, and change the gadgets in the sets  $A^*, B^*$  which ensure that there is an optimal solution that avoids vertices from at most one set  $A'_{i^*}$  and at most one set  $B'_{j^*}$ . Such gadgets have been developed by Lewis and Yannakakis in their generic NP-completeness proof [30]. As their description is somewhat technical, I will not treat them here.

## 5 $H$ -Factor

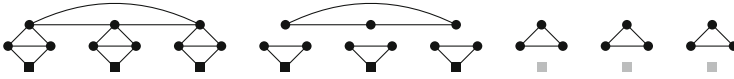
I devote the last case study of this article to generalizations of the MATCHING problem in graphs. For (undirected, simple) graphs  $G$  and  $H$ , an  $H$ -packing in  $G$  is a collection  $H_1, \dots, H_k$  of vertex-disjoint subgraphs of  $G$ , each of which is isomorphic to  $H$ . An  $H$ -factor in  $G$  is an  $H$ -packing  $H_1, \dots, H_k$  whose vertex sets partition  $V(G)$ . The corresponding decision problem  $H$ -FACTOR asks if an input graph  $G$  has an  $H$ -factor. Kirkpatrick and Hell proved [28] that  $H$ -FACTOR is NP-complete when  $H$  contains a connected component of three or more vertices, and is polynomial-time solvable otherwise by a reduction to MAXIMUM MATCHING. Dell and Marx proved [14, Thm. 1.4] under the standard assumption  $\text{NP} \not\subseteq \text{coNP}/\text{poly}$  that for connected graphs  $H$  on at least three vertices, the  $H$ -FACTOR problem parameterized by the number of vertices  $n$  does not admit a generalized kernel of bitsize  $\mathcal{O}(n^{2-\varepsilon})$  for any  $\varepsilon > 0$ . Their proof relies on the Packing Lemma. In this section, I will give an elementary proof of the same theorem. The proof uses the following gadgets by Kirkpatrick and Hell.

**Lemma 2** ([28, Lemma 3.5], cf. [14, Lemma 4.2]). *For each connected graph  $H$  on at least three vertices, there is a graph  $H'$  called the local  $H$ -coordinator gadget, which contains  $|V(H)|$  distinct connector vertices  $C \subseteq V(H')$  as an independent set, and has  $V(H') \setminus C$  as its interior vertices, such that:*

- *There is an  $H$ -factor of  $H'$  and there is an  $H$ -factor of  $H' - C$ .*
- *For each  $\emptyset \subsetneq C' \subsetneq C$  there is no  $H$ -factor of  $H' - C'$ .*
- *The graph  $H' - C$  is connected.*
- *If a graph  $G$  contains  $H'$  as an induced subgraph, such that no interior vertex of  $H'$  is adjacent to a vertex outside of  $H'$ , then in any  $H$ -factor  $H_1, \dots, H_k$  of  $G$ , the following holds: for every interior vertex  $v \in V(H') \setminus C$ , if  $v \in V(H_i)$  then  $V(H_i) \subseteq V(H')$ .*

See Fig. 2 for an example. The last *coherence* property of the gadget effectively ensures that the  $H$ -subgraphs covering interior vertices of  $H'$ , cannot cover any vertices not belonging to  $H'$ .

Consider a fixed graph  $H$  on  $h$  vertices. Given a graph  $G$ , the operation of *attaching a local  $H$ -coordination gadget* onto a set  $S = \{v_1, \dots, v_h\}$  of  $h$  vertices in  $G$  is defined as follows: insert a new disjoint copy of the local coordination gadget  $H'$ , and denote its connector vertices by  $c_1, \dots, c_h$ . For each  $j \in [h]$ , identify  $v_j$  with  $c_j$ , and use  $v_j$  as the identity of the merged vertex. I will use this operation in several constructions.



**Fig. 2.** Left: Local coordination gadget  $H'$  for  $H = K_3$ , whose connector vertices  $C$  are visualized by squares. Middle: a  $K_3$ -factor of  $H'$ . Right: a  $K_3$ -factor of  $H' - C$ .

Lemma 2 yields the following useful property. If a graph  $G$  is obtained by attaching a local  $H$ -coordination gadget  $H'$  onto an independent set  $S$  in an existing graph, and possibly inserting other vertices and edges that are not incident to the interior vertices of  $H'$  in such a way that  $S$  remains an independent set, then in any  $H$ -factor of  $G$  the following holds: either all connector vertices are covered by copies of  $H$  contained entirely within  $H'$ , or no connector vertex is covered by a copy of  $H$  that contains other vertices of  $H'$ . In the former case, I say that the gadget  $H'$  *absorbs* all its connector vertices; in the latter, that the gadget absorbs none of its connector vertices.

As in the earlier sections, a more structured NP-hard version of  $H$ -PACKING is needed as the source problem for the cross-composition. For fixed connected graphs  $H$  and  $F$ , it is defined as follows.

**$H$ -FACTOR WITH  $F$ -PARTITION**

**Input:** A graph  $G$  and a partition of  $V(G)$  into  $A \cup B$  such that  $G[A]$  is an independent set, each connected component of  $G[B]$  is isomorphic to  $F$ , and  $|A|$  and  $|B|$  are both multiples of  $|V(H)|$ .

**Question:** Does  $G$  have an  $H$ -factor?

For each connected graph  $H$  for which  $H$ -FACTOR is NP-hard, there is a connected graph  $F$  for which the above problem is NP-hard;  $F$  can be chosen as the local  $H$ -coordination gadget without its connector vertices. This follows from the construction of Kirkpatrick and Hell [28, Lemma 4.1]. I sketch a proof below, to highlight how the local coordination gadget can be exploited.

**Lemma 3.** *Let  $H$  be a connected graph on at least three vertices, let  $H'$  be the local  $H$ -coordination gadget with connector vertices  $C$  as in Lemma 2, and let  $F := H' - C$ . Then  $H$ -FACTOR WITH  $F$ -PARTITION is NP-hard under Karp reductions.*

*Proof (sketch).* Recall that for every integer  $d \geq 3$ , the PERFECT  $d$ -SET PACKING problem is defined as follows: given a collection of  $S_1, \dots, S_m$  of subsets of size  $d$  of a universe  $U$ , decide whether there exist  $|U|/d$  pairwise disjoint sets in the collection (whose union therefore contains every element of  $U$ ). The PERFECT  $d$ -SET PACKING problem is NP-complete for each  $d \geq 3$  [24, SP1]. To prove the lemma, I show that for  $h := |V(H)|$  there is a Karp reduction from PERFECT  $h$ -SET PACKING to  $H$ -FACTOR WITH  $F$ -PARTITION.

Consider an input  $S_1, \dots, S_m$  over a universe  $U$  for PERFECT  $h$ -SET PACKING. If  $|U|$  is not a multiple of  $h$ , then clearly the answer is NO and we may output

a fixed NO instance. In the remainder, assume  $|U|$  is a multiple of  $h$ . Construct a graph  $G$  as follows. Initialize  $G$  as the edgeless graph on vertex set  $U$ , and attach a local  $H$ -coordination gadget onto the vertices of  $S_i$  for each  $i \in [m]$ . Since Lemma 2 guarantees that the connector vertices form an independent set in  $H'$ , this preserves the fact that  $G[U]$  is edgeless. As  $G - U$  consists of copies of the connected graph  $H' - C = F$ , each connected component of  $G' - U$  is isomorphic to  $F$ . Hence  $(G, A := U, B := V(G) \setminus U)$  is a valid instance of  $H$ -FACTOR WITH  $F$ -PARTITION. Since  $|A| = |U|$ , it is a multiple of  $h = |V(H)|$ . To see that  $|B|$  is a multiple of  $h$ , it suffices to note that  $G[B]$  consists of disjoint copies of  $H' - C$ , each of which has an  $H$ -factor by Lemma 2 and therefore has an integer multiple of  $h$  many vertices. As Lemma 2 guarantees that, for each gadget attached onto a set  $S_i$ , the gadget either absorbs all attached vertices or none, while the interior vertices of gadgets attached for unused sets can be also covered by an  $H$ -factor, it is easy to verify that  $(G, A, B)$  is equivalent to the set packing instance we started from.  $\square$

Lemma 3 provides us with a starting problem for the cross-composition. Before presenting that composition, some more gadgeteering is required. While the local gadget of Lemma 2 synchronizes the behavior of  $|V(H)|$  vertices at a time, in the construction we will need to synchronize the behavior of arbitrarily large vertex sets. For that reason we need a global coordination gadget, which will be described in Lemma 4. The following proposition is needed for its construction.

**Proposition 1.** *There is a polynomial-time algorithm that, given an integer  $h \geq 3$  and an integer  $m \geq 1$ , outputs a connected bipartite multigraph  $F$  with partite sets  $A$  and  $B$ . Set  $A$  has  $m \cdot h$  vertices, each of degree  $h - 1$ , and set  $B$  has  $m \cdot (h - 1)$  vertices, each of degree  $h$ .*

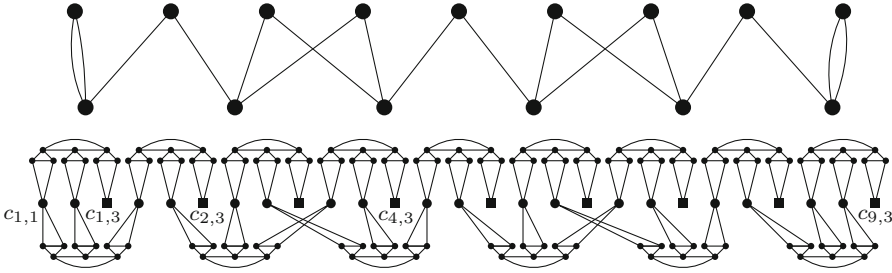
*Proof.* Initialize  $F$  as a cycle on  $2m(h - 1)$  vertices, half of which belong to  $A$  and the other half to  $B$ . Then insert  $m$  additional vertices into  $A$ , each of which is connected by an edge to a distinct vertex of  $B$ . Clearly,  $F$  is a connected bipartite graph with partite sets of the right size. No vertex exceeds its intended degree bound since  $h \geq 3$ . Greedily extend  $F$  to the desired regular bipartite multigraph: as long as there is a vertex in one partite set whose degree is still too small, there is an accompanying vertex in the other partite set whose degree is also too small: insert an edge between them into the multigraph.  $\square$

Now I present the global coordination gadget.

**Lemma 4.** *For each fixed connected graph  $H$  on  $h \geq 3$  vertices, there is a polynomial-time algorithm that, given an integer  $n$  that is a multiple of  $h$ , constructs a graph  $H^*$  on  $\mathcal{O}(n)$  vertices together with an independent set  $C^*$  of  $n$  connector vertices in  $H^*$ , such that:*

1. *There is an  $H$ -factor of  $H^*$  and there is an  $H$ -factor of  $H^* - C^*$ .*
2. *For each  $\emptyset \subsetneq C' \subsetneq C^*$  there is no  $H$ -factor of  $H^* - C'$ .*





**Fig. 3.** Bottom: A global coordination gadget  $H^*$  for  $H = K_3$ , whose  $n = 3 \cdot 3 = 9$  connector vertices  $C^* = \{c_{1,3}, c_{2,3}, \dots, c_{9,3}\}$  are visualized as squares. Top: The connected bipartite multigraph  $F$  whose use in Step 3 of Lemma 4 leads to the bottom gadget  $H^*$ . Observe that  $F$  can be obtained from  $H^*$  by taking one vertex for every local coordination gadget that was inserted, and adding an edge for every pair of gadgets that share a vertex.

3. If a graph  $G$  contains  $H^*$  as an induced subgraph, such that none of the interior vertices  $V(H^*) \setminus C^*$  are adjacent to vertices outside of  $H^*$ , then for every  $H$ -factor  $H_1, \dots, H_k$  of  $G$ , the following holds: if  $v$  is an interior vertex of  $H^*$  and  $v \in H_i$ , then  $V(H_i) \subseteq V(H^*)$ .

*Proof.* Let  $n = m \cdot h$ . We build  $H^*$  as follows. (See Fig. 3 for an illustration.)

1. Initialize  $H^*$  as an independent set on vertex set  $C = \{c_{i,j} \mid i \in [n], j \in [h]\}$ . Define  $C^* := \{c_{i,h} \mid i \in [n]\}$  to be the  $n$  connector vertices.
2. For each  $i \in [n]$ , insert a local  $H$ -coordinator gadget  $\mathcal{A}_i$  into  $H^*$  and attach  $\mathcal{A}_i$  onto  $\{c_{i,j} \mid j \in [h]\}$ . The gadgets  $\mathcal{A}_1, \dots, \mathcal{A}_n$  added in this step are referred to as the *top gadgets*, reflecting their visualization in Fig. 3.
3. Invoke Proposition 1 to construct a connected bipartite multigraph  $F$  with one  $(h - 1)$ -regular partite set  $A = \{a_1, \dots, a_{m \cdot h}\}$ , and one  $h$ -regular partite set  $B = \{b_1, \dots, b_{m \cdot (h-1)}\}$ . Order the edges incident on each vertex  $a_i$  arbitrarily from 1 to  $h - 1$ . Associate to each vertex  $b_k$  a private set of  $h$  vertices from  $C$ : for each edge  $e$  connecting  $b_k$  to a neighbor  $a_i$ , if  $e$  is the  $\ell$ -th incident edge of  $a_i$  in the ordering, then associate  $c_{i,\ell}$  to  $b_k$ . For each  $b_k \in B$ , insert a local  $H$ -coordination gadget  $\mathcal{B}_k$  into  $H^*$  and attach it onto the  $h$  vertices associated to  $b_k$ . This leads to the bottom row of coordinator gadgets in Fig. 3. The regularity conditions of  $F$  ensure that we attach exactly one bottom-row gadget onto each vertex of  $\{c_{i,j} \mid i \in [n], 1 \leq j \leq h - 1\}$ . Moreover, for vertices  $b_k, b_{k'}$  with a common neighbor  $a_i$  in  $F$ , the vertex sets onto which gadgets  $\mathcal{B}_k$  and  $\mathcal{B}_{k'}$  are attached both include a vertex of  $\{c_{i,j} \mid 1 \leq j \leq h - 1\}$ .

It is easy to see that the construction can be carried out in polynomial time and results in a graph on  $\mathcal{O}(n)$  vertices. Note that since  $H$  is fixed, the size of a local coordinator gadget is constant. Let us verify the claimed properties.

(1) To get an  $H$ -factor of  $H^*$ , we combine the  $H$ -factors of the local coordinator gadgets whose existence is guaranteed by Lemma 2, as follows. For each



top-row gadget inserted in Step 2, use an  $H$ -factor of the gadget that absorbs all connector vertices. For each bottom-row gadget inserted in Step 3, use an  $H$ -factor that absorbs no connector vertices.

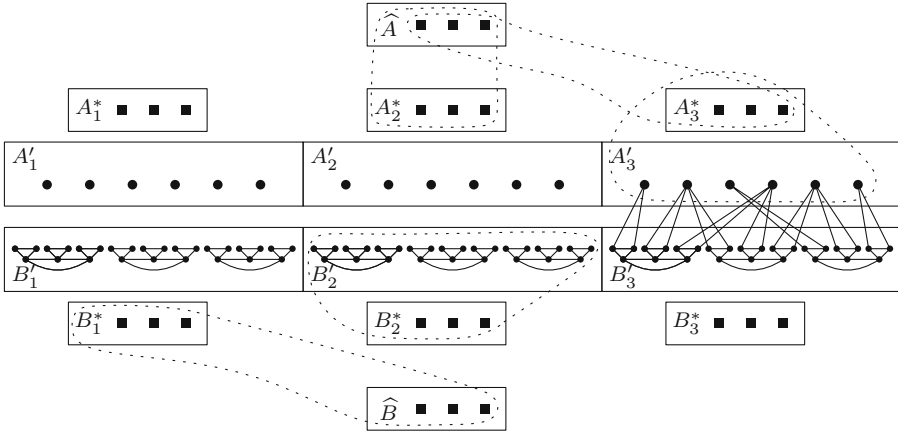
To get an  $H$ -factor of  $H^* - C^*$ , we do the opposite: top-row gadgets inserted in Step 2 absorb no connector vertices, but bottom-row gadgets inserted in Step 3 absorb all connector vertices.

(2) Consider an  $H$ -factor of  $H^* - C'$  for some nonempty  $C' \subseteq C^*$ ; I will show that  $C' = C^*$ . Consider a connector vertex  $c_{i,h} \in C'$  that is not used in the  $H$ -factor of the subgraph. Then the vertices  $X = \{c_{i,j} \mid 1 \leq j \leq h - 1\}$  are not absorbed by the corresponding top-row gadget  $\mathcal{A}_i$  that was attached to  $X \cup \{c_{i,h}\}$ , since Lemma 2 guarantees that  $\mathcal{A}_i$  absorbs either all or none of its local connector vertices. Since graph  $H^* - C'$  contains  $X$  and  $C' \supseteq X$  is an independent set in  $H^*$ , the vertices from  $X$  must therefore be absorbed by bottom-row gadgets in the  $H$ -factor. Let  $\mathcal{B}_k$  be a bottom-row gadget that was attached onto a vertex of  $X$ . Since  $\mathcal{A}_i$  does not absorb its local connector vertices, the vertex shared between  $\mathcal{A}_i$  and  $\mathcal{B}_k$  must be absorbed by  $\mathcal{B}_k$ , which therefore absorbs all its connector vertices. This means that no top-row gadget that shares a vertex with  $\mathcal{B}_k$  can absorb any of its connector vertices. If vertices  $b_k, b_{k'}$  have a common neighbor  $a_{i'}$  in  $F$ , this implies that  $\mathcal{A}_{i'}$  absorbs no connector vertices, so that  $\mathcal{B}_{k'}$  absorbs all its connector vertices. Since  $F$  is connected, repeating this argument shows that all bottom-row gadgets absorb all their local connector vertices, while no top-row gadgets absorb any of their local connector vertices. Consequently, the  $H$ -factor of  $H^* - C'$  does not contain any vertex of  $C^*$  and therefore  $C' = C^*$ .

(3) Suppose that  $H^*$  is contained as an induced subgraph in a larger graph  $G$ , such that no interior vertex is adjacent to a vertex outside  $H^*$ . Consider an  $H$ -factor  $H_1, \dots, H_k$  of  $G$ , and fix an interior vertex  $v$  of  $H^*$ . If  $v$  is an interior vertex of some local coordination gadget, then Lemma 2 ensures that the  $H$ -subgraph  $H_i$  containing  $v$  is contained within the local coordination gadget, and therefore  $V(H_i) \subseteq V(H^*)$ . Otherwise,  $v$  is of the form  $c_{i,j}$  for  $i \in [n]$  and  $j \in [h - 1]$ . But then the only neighbors of  $v$  in  $G$  are interior vertices of local coordination gadgets inserted into  $H^*$ . Since  $H$  is connected and has at least three vertices, vertex  $v$  is contained in some  $H$ -subgraph  $H_i$  together with an internal vertex of a local coordination gadget, ensuring  $V(H_i) \subseteq V(H^*)$  by the previous argument. This concludes the proof.  $\square$

Using the properties of Lemma 4, the terminology of attaching coordination gadgets and absorbing connector vertices extends to global coordination gadgets in the natural way. The sparsification lower bound for  $H$ -FACTOR now follows cleanly by combining the two ingredients developed so far: the “bipartite” NP-hard source problem of Lemma 3 and the global coordination gadget of Lemma 4.

**Theorem 4.** *For any  $\varepsilon > 0$ , for any connected graph  $H$  on at least three vertices,  $H$ -FACTOR parameterized by the number of vertices  $n$  does not admit a generalized kernelization of bitsize  $\mathcal{O}(n^{2-\varepsilon})$  unless  $\text{NP} \subseteq \text{coNP}/\text{poly}$ .*



**Fig. 4.** Schematic visualization of the result of the cross-composition of Theorem 4 for  $H = K_3$ , applied to  $t = 9$  inputs of  $H$ -FACTOR WITH  $F$ -PARTITION. Of the edges between different sets  $A'_i$  and  $B'_j$ , only those corresponding to the YES-instance induced by  $A'_3$  and  $B'_3$  have been drawn. Vertices of inserted global coordination gadgets are not drawn. For some global coordination gadgets, the vertices they have been attached onto have been highlighted by a dotted curve.

*Proof.* I present a degree-2 OR-cross-composition. Fix a graph  $F$  such that the source problem  $H$ -FACTOR WITH  $F$ -PARTITION is NP-hard. Let  $(G_{i,j}, A_{i,j}, B_{i,j})$  for  $i, j \in [\sqrt{t}]$  be a sequence of  $t$  input instances that all share the same number  $n_A$  of vertices in the  $A$ -set and the same number  $n_B$  of vertices in the  $B$ -set, which may be assumed by a suitable choice of  $\mathcal{R}$ . By definition of the source problem, both  $n_A$  and  $n_B$  are multiples of  $h = |V(H)|$ . Build an instance  $G'$  of  $H$ -FACTOR; refer to Fig. 4 for an illustration.

1. For each  $i \in [\sqrt{t}]$ , add a set  $A'_i$  of  $n_A$  independent vertices to  $G'$ .
2. For each  $i \in [\sqrt{t}]$ , add a set  $B'_i$  of  $n_B$  vertices to  $G'$ . Insert edges so that  $G'[B'_i]$  forms  $n_B/|V(F)|$  vertex-disjoint copies of  $F$ .
3. For each  $i, j \in [\sqrt{t}]$ , insert edges between  $A'_i$  and  $B'_j$  so that  $G'[A'_i \cup B'_j]$  is isomorphic to  $G_{i,j}$ . This is simultaneously possible for all  $i, j$  since all graphs  $(G_{i,j}[A_{i,j}])_{i,j \in [\sqrt{t}]}$  are isomorphic to each other, and similarly all graphs  $(G_{i,j}[B_{i,j}])_{i,j \in [\sqrt{t}]}$  are isomorphic to each other. Since each connected component of  $G_{i,j}[B_{i,j}]$  is isomorphic to the fixed graph  $F$ , this step can be performed in polynomial time.
4. For each  $i \in [\sqrt{t}]$ , add a vertex set  $A_i^*$  of size  $h$  to  $G'$ , and attach a new global coordination gadget  $A_i$  with  $n_A + h$  connector vertices onto  $A'_i \cup A_i^*$ .
5. For each  $j \in [\sqrt{t}]$ , add a vertex set  $B_j^*$  of size  $h$  to  $G'$ , and attach a new global coordination gadget  $B_j$  with  $n_B + h$  connector vertices onto  $B'_j \cup B_j^*$ .
6. Add a vertex set  $\hat{A}$  of size  $h$  to  $G'$ . For each  $i \in [\sqrt{t}]$ , insert a global coordination gadget  $\hat{A}_i$  with  $2h$  connector vertices, and attach it onto  $\hat{A} \cup A_i^*$ .

7. Add a vertex set  $\widehat{B}$  of size  $h$  to  $G'$ . For each  $j \in [\sqrt{t}]$ , insert a global coordination gadget  $\widehat{B}_j$  with  $2h$  connector vertices, and attach it onto  $\widehat{B} \cup B_j^*$ .

This concludes the description of graph  $G'$ . It is easy to see that the construction can be performed in polynomial time. Let us analyze the number of vertices in  $G'$ . It is easy to verify that, apart from the vertices of the inserted global coordination gadgets, the graph has  $\mathcal{O}(\sqrt{t}(n_A + n_B))$  vertices, treating  $|V(H)|$  as a constant. In Steps 4–5 we insert  $\mathcal{O}(\sqrt{t})$  global coordination gadgets for  $\mathcal{O}(n_A + n_B)$  connector vertices each, which therefore contribute  $\mathcal{O}(\sqrt{t}(n_A + n_B))$  vertices to  $G'$ . Finally, the last two steps contribute  $\mathcal{O}(\sqrt{t})$  additional vertices. It follows that  $|V(G')| \in \mathcal{O}(\sqrt{t}(n_A + n_B))$ , which is suitably bounded.

To complete the cross-composition, we verify that  $G'$  has an  $H$ -factor if and only if some input instance  $G_{i^*,j^*}$  has an  $H$ -factor.

Suppose first that  $G_{i^*,j^*}$  has an  $H$ -factor. Since  $G'[A_{i^*}^* \cup B_{j^*}^*]$  is isomorphic to  $G_{i^*,j^*}$ , it has an  $H$ -factor. To extend it to an  $H$ -factor of all of  $G'$ , we do the following. For each  $i \in [\sqrt{t}] \setminus \{i^*\}$ , use an  $H$ -factor of the gadget  $\mathcal{A}_i$  together with its connector vertices, absorbing  $A_i^* \cup A_i^*$ . Similarly, for each  $j \in [\sqrt{t}] \setminus \{j^*\}$ , use an  $H$ -factor of  $\mathcal{B}_j$  together with its connector vertices, absorbing  $B_j^* \cup B_j^*$ . Use an  $H$ -factor of gadget  $\widehat{\mathcal{A}}_{i^*}$  with its connector vertices, absorbing  $A_{i^*}^* \cup \widehat{A}$ . Similarly, use an  $H$ -factor of gadget  $\widehat{\mathcal{B}}_{j^*}$  with its connector vertices, absorbing  $B_{j^*}^* \cup \widehat{B}$ . For the remaining global coordination gadgets, use  $H$ -factors of only the interior of the gadget without absorbing connector vertices. This yields an  $H$ -factor of  $G'$ .

For the other direction, suppose that  $G'$  has an  $H$ -factor. Since the vertices of  $\widehat{A}$  and  $\widehat{B}$  form independent sets, whose only interaction with the rest of the graph is through the attachment of coordination gadgets, using Lemma 4 it follows there is a global coordination gadget  $\widehat{\mathcal{A}}_{i^*}$  that absorbs all its connection vertices  $A_{i^*}^* \cup \widehat{A}$ , and analogously a gadget  $\widehat{\mathcal{B}}_{j^*}$  absorbing  $B_{j^*}^* \cup \widehat{B}$ . Consequently, for each  $i \neq i^*$  the vertices of  $A_i^*$  are absorbed by the global coordination gadget  $\mathcal{A}_i$ , which therefore also absorbs  $A_i^*$ . Similarly, for each  $j \neq j^*$  the vertices of  $B_j^*$  are absorbed by  $\mathcal{B}_j$ , thereby also absorbing  $B_j^*$ . This implies that in the  $H$ -factor of  $G'$ , vertices of  $A_{i^*}^* \cup B_{j^*}^*$  are not contained in  $H$ -subgraphs together with vertices outside of  $A_{i^*}^* \cup B_{j^*}^*$ . Consequently, the  $H$ -factor of  $G'$  restricted to  $A_{i^*}^* \cup B_{j^*}^*$  is an  $H$ -factor of  $G'[A_{i^*}^* \cup B_{j^*}^*]$ , which is isomorphic to  $G_{i^*,j^*}$ . Hence  $G_{i^*,j^*}$  is a YES-instance. This concludes the proof of Theorem 4.  $\square$

Theorem 4 shows that for *connected* graphs  $H$  on at least three vertices, the  $H$ -FACTOR problem admits no nontrivial polynomial-time sparsification unless  $\text{NP} \subseteq \text{coNP/poly}$ . It is not difficult to extend this to disconnected graphs  $H$  that have a connected component of at least three vertices, thereby extending the lower bound to all graphs  $H$  for which  $H$ -FACTOR is NP-hard.

**Corollary 1.** *For any  $\varepsilon > 0$ , for any graph  $H$  that contains a connected component on at least three vertices,  $H$ -FACTOR parameterized by the number of vertices  $n$  does not admit a generalized kernelization of bitsize  $\mathcal{O}(n^{2-\varepsilon})$  unless  $\text{NP} \subseteq \text{coNP/poly}$ .*

*Proof.* Consider such a graph  $H$ , and let  $H'$  be a connected component of  $H$  on at least three vertices that maximizes the number of edges. Using a reduction of Kirkpatrick and Hell [28, Lemma 2.1], I give a polynomial-time reduction from  $H'$ -FACTOR instances on  $n$  vertices to  $H$ -FACTOR instances on  $\mathcal{O}(n)$  vertices. This reduction, together with a subquadratic generalized kernelization for  $H$ -FACTOR, would yield a subquadratic generalized kernelization for  $H'$ -FACTOR, which is ruled out by Theorem 4. Hence it suffices to give the reduction.

Given a graph  $G'$  for which we want to determine the existence of an  $H'$ -factor, we may assume without loss of generality that  $|V(G')|$  is a multiple of  $|V(H')|$  (otherwise the answer is trivially NO). If  $|V(G')| = d \cdot |V(H')|$ , then for the graph  $G$  obtained as the disjoint union of  $G'$  together with  $d$  copies of  $H - H'$ , it is easy to verify (cf. [28, Lemma 2.1]) that  $G'$  has an  $H'$ -factor if and only if  $G$  has an  $H$ -factor. Since  $H$  is fixed,  $|V(G')| \in \mathcal{O}(|V(G)|)$ , which concludes the proof.  $\square$

## 6 Conclusion

In this article, I showed how the technique of cross-composition [7] that was developed together with Hans Bodlaender and Stefan Kratsch can be used to give elementary proofs that VERTEX COVER, FEEDBACK VERTEX SET, and  $H$ -PACKING do not admit generalized kernels of bitsize  $\mathcal{O}(n^{2-\varepsilon})$ . The constructions all boil down to the appropriate combination of two key ingredients: a suitable NP-hard starting problem whose inputs can be partitioned into two regularly structured induced subgraphs, and a gadget that allows some parts of the input to be disabled in order to achieve the desired logical OR.

Following the case-by-case investigation of sparsification lower bounds in this article, one is naturally led to ask whether it is possible to prove sparsification lower bounds on a larger scale, capturing entire classes of problems at the same time. In recent years, I have pursued this direction together with my PhD student Astrid Pieterse (Hans' academic granddaughter!) by investigating the sparsifiability of CONSTRAINT SATISFACTION problems. A key challenge for the future consist of the following: for which constraint languages (defining the types of constraints that one is allowed to use in the problem) do CSPs over the Boolean domain allow for a linear sparsification? I refer the interested reader to recent papers [12, 29] for further information.

An entirely different, but equally exciting, direction is the study of the positive toolkit: sparsification algorithms. While nontrivial sparsification algorithms exist for several satisfiability problems, such as 3-NOT ALL EQUAL SATISFIABILITY parameterized by the number of variables  $n$ , which can be sparsified [26] to instances with  $\mathcal{O}(n^2)$  clauses, to this day we do not have any good examples of NP-hard *graph* problems that admit nontrivial polynomial-time sparsification. Have we not been looking in the right places, or could there be a reason for their nonexistence?

I conclude this article by thanking Hans; for saving me from becoming a computer-game programmer; for inspiring me to do research; and for his

good-humored companionship, discussions, and boardgames spanning over a decade and multiple continents. Happy 60th birthday, Hans!

## References

1. Bodlaender, H.L.: A cubic kernel for feedback vertex set. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 320–331. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-70918-3\\_28](https://doi.org/10.1007/978-3-540-70918-3_28)
2. Bodlaender, H.L., van Dijk, T.C.: A cubic kernel for feedback vertex set and loop cutset. *Theory Comput. Syst.* **46**(3), 566–597 (2010). <https://doi.org/10.1007/s00224-009-9234-2>
3. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels (extended abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5125, pp. 563–574. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-70575-8\\_46](https://doi.org/10.1007/978-3-540-70575-8_46)
4. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *J. Comput. Syst. Sci.* **75**(8), 423–434 (2009). <https://doi.org/10.1016/j.jcss.2009.04.001>
5. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Cross-composition: a new technique for kernelization lower bounds. In: Schwentick, T., Dürr, C. (eds.) Proceedings of the 28th STACS. LIPIcs, vol. 9, pp. 165–176. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011). <https://doi.org/10.4230/LIPIcs.STACS.2011.165>
6. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernel bounds for path and cycle problems. *Theor. Comput. Sci.* **511**, 117–136 (2013). <https://doi.org/10.1016/j.tcs.2012.09.006>
7. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.* **28**(1), 277–305 (2014). <https://doi.org/10.1137/120880240>
8. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.* **412**(35), 4570–4578 (2011). <https://doi.org/10.1016/j.tcs.2011.04.039>
9. Brandt, S.: Computing the independence number of dense triangle-free graphs. In: Möhring, R.H. (ed.) WG 1997. LNCS, vol. 1335, pp. 100–108. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0024491>
10. Burrage, K., Estivill-Castro, V., Fellows, M., Langston, M., Mac, S., Rosamond, F.: The undirected feedback vertex set problem has a poly( $k$ ) kernel. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 192–202. Springer, Heidelberg (2006). [https://doi.org/10.1007/11847250\\_18](https://doi.org/10.1007/11847250_18)
11. Buss, J.F., Goldsmith, J.: Nondeterminism within P. *SIAM J. Comput.* **22**(3), 560–572 (1993). <https://doi.org/10.1137/0222038>
12. Chen, H., Jansen, B.M.P., Pieterse, A.: Best-case and worst-case sparsifiability of Boolean CSPs. In: Paul, C., Pilipczuk, M. (eds.) Proceedings of 13th IPEC. LIPIcs, vol. 115, pp. 15:1–15:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018). <https://doi.org/10.4230/LIPIcs.IPEC.2018.15>
13. Cygan, M., et al.: *Parameterized Algorithms*. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-21275-3>
14. Dell, H., Marx, D.: Kernelization of packing problems. In: Rabani, Y. (ed.) Proceedings of the 23rd SODA, pp. 68–81. SIAM (2012). <https://doi.org/10.1137/1.9781611973099.6>

15. 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 STOC, pp. 251–260. ACM (2010). <https://doi.org/10.1145/1806689.1806725>
16. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM* **61**(4), 23:1–23:27 (2014). <https://doi.org/10.1145/2629620>
17. Dom, M., Lokshtanov, D., Saurabh, S.: Kernelization lower bounds through colors and IDs. *ACM Trans. Algorithms* **11**(2), 13 (2014). <https://doi.org/10.1145/2650261>
18. Downey, R., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999). <https://doi.org/10.1007/978-1-4612-0515-9>
19. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, London (2013). <https://doi.org/10.1007/978-1-4471-5559-1>
20. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006). <https://doi.org/10.1007/3-540-29953-X>
21. Fomin, F.V., Lokshtanov, D., Saurabh, S., Zehavi, M.: Kernelization: Theory of Parameterized Preprocessing. Cambridge University Press, Cambridge (2019). <https://doi.org/10.1017/9781107415157>
22. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. In: Dwork, C. (ed.) Proceedings of the 40th STOC, pp. 133–142. ACM (2008). <https://doi.org/10.1145/1374376.1374398>
23. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.* **77**(1), 91–106 (2011). <https://doi.org/10.1016/j.jcss.2010.06.007>
24. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
25. Iwata, Y.: Linear-time kernelization for feedback vertex set. In: Chatzigiannakis, I., Indyk, P., Kuhn, F., Muscholl, A. (eds.) Proceedings of the 44th ICALP. LIPIcs, vol. 80, pp. 68:1–68:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017). <https://doi.org/10.4230/LIPIcs.ICALP.2017.68>
26. Jansen, B.M.P., Pieterse, A.: Optimal sparsification for some binary CSPs using low-degree polynomials. *TOCT* **11**(4), 28:1–28:26 (2019). <https://doi.org/10.1145/3349618>
27. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Springer, Boston (1972). [https://doi.org/10.1007/978-1-4684-2001-2\\_9](https://doi.org/10.1007/978-1-4684-2001-2_9)
28. Kirkpatrick, D.G., Hell, P.: On the complexity of general graph factor problems. *SIAM J. Comput.* **12**(3), 601–609 (1983). <https://doi.org/10.1137/0212040>
29. Lagerkvist, V., Wahlström, M.: Kernelization of constraint satisfaction problems: a study through universal algebra. In: Beck, J.C. (ed.) CP 2017. LNCS, vol. 10416, pp. 157–171. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-66158-2\\_11](https://doi.org/10.1007/978-3-319-66158-2_11)
30. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.* **20**(2), 219–230 (1980). [https://doi.org/10.1016/0022-0000\(80\)90060-4](https://doi.org/10.1016/0022-0000(80)90060-4)
31. Nemhauser, G., Trotter, L.: Vertex packings: structural properties and algorithms. *Math. Program.* **8**, 232–248 (1975). <https://doi.org/10.1007/BF01580444>

32. Poljak, S.: A note on stable sets and colorings of graphs. *Commentationes Mathematicae Universitatis Carolinae* **015**(2), 307–309 (1974). <http://eudml.org/doc/16622>
33. Thomassé, S.: A  $4k^2$  kernel for feedback vertex set. *ACM Trans. Algorithms* **6**(2) (2010). <https://doi.org/10.1145/1721837.1721848>

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

