Solving Partition Problems Almost Always Requires Pushing Many Vertices Around

Iyad Kanj
School of Computing, DePaul University Chicago, USA
ikanj@cs.depaul.edu

Christian Komusiewicz[1]
Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany
komusiewicz@informatik.uni-marburg.de

Manuel Sorge[2]
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer Sheva, Israel
sorge@post.bgu.ac.il

Erik Jan van Leeuwen
Department of Information and Computing Sciences, Utrecht University, The Netherlands
e.j.vanleeuwen@uu.nl

Abstract

A fundamental graph problem is to recognize whether the vertex set of a graph G can be bipartitioned into sets A and B such that $G[A]$ and $G[B]$ satisfy properties Π_A and Π_B, respectively. This so-called (Π_A, Π_B)-Recognition problem generalizes amongst others the recognition of 3-colorable, bipartite, split, and monopolar graphs. A powerful algorithmic technique that can be used to obtain fixed-parameter algorithms for many cases of (Π_A, Π_B)-Recognition, as well as several other problems, is the *pushing process*. For bipartition problems, the process starts with an "almost correct" bipartition (A', B'), and pushes appropriate vertices from A' to B' and vice versa to eventually arrive at a correct bipartition.

In this paper, we study whether (Π_A, Π_B)-Recognition problems for which the pushing process yields fixed-parameter algorithms also admit polynomial problem kernels. In our study, we focus on the first level above triviality, where Π_A is the set of P_3-free graphs (disjoint unions of cliques, or cluster graphs), the parameter is the number of clusters in the cluster graph $G[A]$, and Π_B is characterized by a set $\mathcal{H}$ of connected forbidden induced subgraphs. We prove that, under the assumption that NP $\not\subseteq$ coNP/poly, (Π_A, Π_B)-Recognition admits a polynomial kernel if and only if $\mathcal{H}$ contains a graph of order at most 2. In both the kernelization and the lower bound results, we make crucial use of the pushing process.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms, Theory of computation → Algorithm design techniques

Keywords and phrases Fixed-parameter algorithms, Kernelization, Vertex-partition problems, Reduction rules, Cross-composition

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.51

[1] CK gratefully acknowledges support by the DFG, project MAGZ, KO 3669/4-1.
[2] MS gratefully acknowledges support by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11, by the Israel Science Foundation (grant number 551145/14), and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement number 714704.

1 Introduction

A graph G is a (Π_A, Π_B)-*graph*, for two hereditary graph properties Π_A, Π_B, if $V(G)$ can be partitioned into two sets A, B such that $G[A] \in \Pi_A$ and $G[B] \in \Pi_B$. We call (A, B) a (Π_A, Π_B)-*partition* of G. The (Π_A, Π_B)-Recognition problem is to recognize whether a given graph is a (Π_A, Π_B)-graph. This captures a wealth of famous problems, including the recognition of 3-colorable, bipartite, co-bipartite, and split graphs, and Π-Vertex Deletion, which asks for a partition (A, B) such that $G[A] \in \Pi$ and $G[B]$ has order at most k for some given k. In the most interesting (and NP-hard) cases [2, 13, 22], Π_A and Π_B are both characterized by a (not necessarily finite) set of forbidden connected induced subgraphs. In other words, Π_A and Π_B are each closed under the disjoint union of graphs in these cases.

Many such (Π_A, Π_B)-Recognition problems were shown fixed-parameter tractable by Kanj *et al.* [20], for example when Π_A is the class of graphs that is a disjoint union of k cliques, using parameter k. The central algorithmic idea that was employed in [20] is the *pushing process*. The algorithm empties the input graph, and adds vertices back one by one while maintaining a valid partition. Since adding a vertex might invalidate a previously valid partition, vertices are *pushed* from one part of the partition to the other part in the hope of obtaining a valid partition again. A similar algorithmic idea, known as iterative localization, was used earlier by Heggernes *et al.* [19] to show the fixed-parameter tractability of computing the cochromatic number of perfect graphs and the stabbing number of disjoint rectangles with axes-parallel lines (using the standard parameters). Iterative localization was also applied in follow-up work related to the cochromatic number [21].

A crucial ingredient in applying the pushing process is to understand the *avalanches* caused by this process. For (Π_A, Π_B)-Recognition, an avalanche is triggered when a vertex is pushed to A; this may imply that several other vertices must be pushed to B, which, in turn, triggers the pushing of yet more vertices to A, and so on. Similar effects are visible in the aforementioned cochromatic number and rectangle stabbing number problems. The contribution of the previous works [19, 20, 21] was to bound the depth of this process by some function of the parameter, leading to fixed-parameter algorithms. However, such a bound does not provide an answer to the question of which vertices trigger avalanches and their continued rolling, and whether the number of such vertices can somehow be limited.

This question can be naturally formalized in terms of the kernelization complexity of problems to which the pushing process applies. A kernel reduces the size of the graph and thus directly reduces the number of vertices triggering or being affected by avalanches when an algorithm based on the pushing process is applied to the kernelized instance. In previous work, Kolay *et al.* [21] studied the kernelization complexity of computing the cochromatic number of a perfect graph G, which is the smallest number $k = r + \ell$ such that $V(G)$ can be partitioned into r sets that each induces a clique and ℓ sets that each induces an edgeless graph. This problem has a parameterized algorithm using iterative localization (*i.e.*, a pushing process) [19], but Kolay *et al.* [21] showed that, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, this problem does not admit a polynomial kernel parameterized by $r + \ell$. This suggests that, for this problem, one cannot control the number of vertices affected by avalanches. The kernelization complexity of (Π_A, Π_B)-Recognition, however, has not been studied so far. Hence, it is open whether avalanches can be controlled to affect few vertices in this case.

Our Result. We study the kernelization complexity of (Π_A, Π_B)-Recognition through the lens of the pushing process. To this end, we consider the first level above triviality of the problem. When Π_A is characterized by a forbidden induced subgraph of order 2, then

(Π_A, Π_B)-Recognition can be solved in linear time [16], and thus we focus on the NP-hard case when the forbidden induced subgraph has order 3 [2, 13, 22]. In particular, we let Π_A be the class of so-called *cluster graphs*. These are the graphs that contain no P_3 – the (simple) path on three vertices – as an induced subgraph, or equivalently, graphs that are disjoint unions of complete graphs. This leads to the following problem:

Cluster-Π-Partition
Input: A graph $G = (V, E)$.
Question: Is there a partition (A, B) of V such that $G[A]$ is a cluster graph and $G[B] \in \Pi$?

Cluster-Π-Partition generalizes the recognition problem of many graph classes, such as the recognition of *monopolar graphs* [6, 9, 8, 23] (Π is the set of edgeless graphs), 2-subcolorable graphs [5, 15, 18, 24] (Π is the set of cluster graphs), and several others [1, 4, 7]. Unfortunately, Cluster-Π-Partition is NP-hard in these special cases, and in general when Π is characterized by a set of connected forbidden induced subgraphs [2, 13, 22]. Hence, we consider the number k of clusters in the cluster graph $G[A]$ as a parameter, and study the pushing process with respect to this parameter.

Our result gives a complete characterization of the kernelization complexity of Cluster-Π-Partition through a deeper understanding of the pushing process. We show that, while for a specific Π the pushing process can be used to witness a small vertex set of size $k^{O(1)}$ containing the vertices affected by avalanches, for all other Π, such a set of polynomial size is unlikely to exist. Formally, we show that:

▶ **Theorem 1.1.** *Let Π be a graph property characterized by a (not necessarily finite) set $\mathcal{H}$ of connected forbidden induced subgraphs. Then unless* NP $\subseteq$ coNP/poly, Cluster-Π-Partition *parameterized by the number k of clusters in the cluster graph $G[A]$ admits a polynomial kernel if and only if $\mathcal{H}$ contains a graph of order at most* 2.

The positive result corresponds to the recognition of monopolar graphs. Indeed, the graph properties with forbidden induced subgraphs of order 2 are "being edgeless" and "being nonedge-less", but the latter is not characterized by connected forbidden induced subgraphs.

The pushing process and a deeper understanding of the avalanches it causes are indeed central to both directions of the above result. In the proof of the positive result, we first perform a set of data reduction rules to identify some vertices that are part of A or B in *any* partition (A, B) of $V(G)$ such that $G[A]$ is a cluster graph with at most k clusters and $G[B]$ is edgeless. More importantly, these rules restrict the combinatorial properties of the graph induced by the remaining vertices. With these restrictions, it becomes possible to model the avalanches that occur using a bipartite graph. This graph enables two further reduction rules that lead to the polynomial kernel.

For the negative result, we observe that the bipartite graph constructed in the kernel is closely tied to the deterministic behavior of the pushing process for monopolar graphs: when an edge in $G[B]$ is created by pushing a vertex to B, the other endpoint of the edge must be pushed to A (recall that $G[B]$ must become edgeless). This limits the avalanches. However, for more complex properties Π_B, such a simple correspondence no longer exists. In particular, when the forbidden induced subgraphs have order at least 3, pushing a vertex to B may create a forbidden induced subgraph in $G[B]$ that can be repaired in at least two different ways. Then the pushing process starts to behave nondeterministically, and the avalanches grow beyond control. We exploit this intuition to exclude the existence of a polynomial kernel, unless NP $\subseteq$ coNP/poly, by providing a cross-composition.

Other Parameterizations. One might consider two other parameters: the size of a largest cluster in $G[A]$ and the size of one of the sides. The size of a largest cluster in $G[A]$ will not lead to tractability, as Cluster-Π-Partition is NP-hard on subcubic graphs, even when Π is the set of edgeless graphs [23]. Thus, we consider the number k of vertices in the graph $G[B]$, even for the broader (Π_A, Π_B)-Recognition problem, observing a general result:

▶ **Theorem 1.2.** (♠)[3] (Π_A, Π_B)-Recognition *has a kernel of size* $\mathcal{O}(k^d)$ *parameterized by* k*, the maximum size of* B*, when* Π_A *can be characterized by a collection* $\mathcal{H}$ *of forbidden induced subgraphs, each of size at most* d*, and* Π_B *is hereditary.*

We obtain the following better bound in terms of the number of vertices for Cluster-Π_Δ-Partition, the restriction of Cluster-Π-Partition to the case when all graphs containing a vertex of degree at least $\Delta + 1$ are forbidden induced subgraphs of Π.

▶ **Theorem 1.3.** (♠) Cluster-Π_Δ-Partition *parameterized by* k*, the maximum size of* B*, has an* $\mathcal{O}((\Delta^2 + 1) \cdot k^2)$*-vertex kernel.*

Preliminaries. We follow standard graph-theoretic notation [11]. For $\ell \in \mathbb{N}$, we use $[\ell]$ to denote $\{1, 2, \ldots, \ell\}$. Let $v \in V(G)$ and $X, Y \subseteq V(G)$. We say v is *adjacent to* X if v is adjacent to at least one vertex in X. We say X *is adjacent to* Y if there exists $x \in X$ that is adjacent to Y. We say a partition (A, B) of $V(G)$ is a *cluster*-Π *partition* if (1) $G[A]$ is a cluster graph and (2) $G[B] \in \Pi$. A *monopolar partition* of a graph G is a partition of $V(G)$ into a cluster graph and an independent set. Monopolar Recognition asks, given a graph G and an integer k, whether G admits a monopolar partition (A, B) such that the number of clusters in the cluster graph $G[A]$ is at most k. For an instance (G, k) of Monopolar Recognition, a monopolar partition of G is *valid* if the number of clusters in the cluster graph of the partition is at most k. For relevant definitions of parameterized complexity, e.g. polynomial problem kernels, see [12, 10]. Let Q be a language and (P, κ) a parameterized problem, *i.e.*, P is a language and $\kappa \colon \Sigma^* \to \mathbb{N}$ a parameterization. An *or-cross-composition* from Q into (P, κ) is a polynomial-time algorithm that, given t instances $q_1, \ldots, q_t \in \Sigma^*$ of Q, computes an instance $r \in \Sigma^*$ such that $\kappa(r) \leq \text{poly}\left(\log t + \max_{i=1}^{t} |q_i|\right)$, and $r \in P$ if and only if $q_i \in Q$ for some $i \in [t]$. If there is an or-cross-composition from an NP-hard language into (P, κ), then there is no polynomial-size problem kernel for (P, κ) unless NP $\subseteq$ coNP/poly [17, 3].

2 A Polynomial Kernel for Monopolar Recognition Parameterized by the Number of Clusters

The outline of the kernelization algorithm is as follows. First, we compute a decomposition of the input graph into sets of vertex-disjoint maximal cliques which we call a *clique decomposition*. This decomposition is used and updated throughout the data-reduction procedure. We also maintain sets of vertices that are determined to belong to A or B. We first apply a sequence of reduction rules whose aim is roughly to bound the number of cliques and the number of edges between the cliques in the decomposition, and to restrict the structure of edges between cliques. Then, we build an auxiliary graph to model how the placement of a vertex in A or B implies an avalanche of placements of vertices in A and B. If this avalanche creates too many clusters in A, then this determines the placement of certain

[3] Due to lack of space, proofs of statements marked with (♠) are omitted.

vertices in A or B, and triggers another reduction rule. If this reduction rule does not apply anymore, then the size of the auxiliary graph is bounded, which in turn, helps bounding the size of the instance.

Clique Decompositions. Say that a clique C is a *large clique* if $|C| \geq 3$, an *edge clique* if $|C| = 2$ (*i.e.*, C is an edge), and a *vertex clique* if $|C| = 1$ (*i.e.*, C consists of a single vertex). Let (G, k) be an instance of Monopolar Recognition. Suppose that $A_{\mathsf{true}} \subseteq V(G)$ and $B_{\mathsf{true}} \subseteq V(G)$ are subsets of vertices that have been determined to be in A and B, respectively, in any valid monopolar partition of (G, k). We define a decomposition $(C_1, \ldots, C_r)$ of $V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$, referred to as a *nice clique decomposition*, that partitions this set into vertex-disjoint cliques $C_1, \ldots, C_r$, $r \geq 1$, such that the tuple $(C_1, \ldots, C_r)$ satisfies the following properties:

- **(i)** In the decomposition tuple $(C_1, \ldots, C_r)$, the large cliques appear before the edge cliques, and the edge cliques, in turn, appear before the vertex cliques; that is, for each large clique C_i and for each edge or vertex clique C_j we have $i < j$, and for each edge clique C_i and for each vertex clique C_j we have $i < j$.
- **(ii)** Each clique C_i, $i \in [r-1]$, is maximal in $\bigcup_{j=i}^{r} C_j$; that is, there does not exist a vertex $v \in \bigcup_{j=i+1}^{r} C_j$ such that $C_i \cup \{v\}$ is a clique.
- **(iii)** The subgraph of G induced by the union of the edge cliques and vertex cliques does not contain any large clique.

The following fact is implied by property (ii) above:

▶ **Fact 2.1.** *The vertex cliques in a nice clique decomposition form an independent set in G.*

▶ **Lemma 2.2.** (♠) *A nice clique decomposition of G can be computed in $\mathcal{O}(nm)$ time.*

Let (G, k) be an instance of Monopolar Recognition. We initialize $A_{\mathsf{true}} = B_{\mathsf{true}} = \emptyset$, $V' = V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$, and we compute a nice clique decomposition $(C_1, \ldots, C_r)$ of V'. We will then apply reduction rules to simplify the instance (G, k). During this process, we may identify vertices in V' to be added to A_{true} or B_{true}. At any point in the process, we will maintain a partition $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ of $V(G)$ such that (1) $A_{\mathsf{true}} \subseteq A$ and $B_{\mathsf{true}} \subseteq B$ for any valid monopolar partition (A, B) of $V(G)$, and (2) $(C_1, \ldots, C_r)$ is a nice clique decomposition of $V' = V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$; we call such a partition $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ a *normalized partition* of $V(G)$.

Basic Reduction Rules. We now describe our basic set of reduction rules. After the application of a reduction rule, a normalized partition may change as the result of moving vertices from $\bigcup_{i=1}^{r} C_i$ to $A_{\mathsf{true}} \cup B_{\mathsf{true}}$, and we will need to compute a nice clique decomposition of the resulting (new) set $V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$. However, a vertex that has been moved to A_{true} (resp. B_{true}) will remain in A_{true} (resp. B_{true}). When a reduction rule is applied, we assume that no reduction rule preceding it is applicable. The following rule is straightforward:

▶ **Reduction Rule 2.3.** *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. If A_{true} is not a cluster graph with at most k clusters, or B_{true} is not an independent set, then reject the instance (G, k).*

The following rule is correct because, for every monopolar partition (A, B) of G, $B_{\mathsf{true}} \subseteq B$ and B is an independent set.

▶ **Reduction Rule 2.4.** *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. If there is a vertex $v \in V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$ that is adjacent to B_{true} then set $A_{\mathsf{true}} = A_{\mathsf{true}} \cup \{v\}$.*

The following rule is correct, since $A_{\mathsf{true}} \subseteq A$ for every monopolar partition (A, B) of G:

▶ **Reduction Rule 2.5.** *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. If there is a vertex $v \in V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$ that is either (1) adjacent to two clusters in A_{true}, or (2) adjacent to a cluster C in A_{true} but not to all the vertices in C, then set $B_{\mathsf{true}} = B_{\mathsf{true}} \cup \{v\}$.*

The proof of the following reduction rule is straightforward, after recalling that the vertex cliques induce an independent set in G (Fact 2.1), and observing that no two vertices of an independent set can belong to the same cluster in a cluster graph:

▶ **Reduction Rule 2.6.** *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. If there is a vertex $v \in V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$ with more than k neighbors that are vertex cliques, then set $A_{\mathsf{true}} = A_{\mathsf{true}} \cup \{v\}$.*

The next two reduction rules restrict the number and type of edges incident to large cliques.

▶ **Reduction Rule 2.7.** *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. If there exists a vertex $v \in V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$ and a large clique C_i such that $1 < |N(v) \cap C_i| \leq |C_i| - 1$, then set $A_{\mathsf{true}} = A_{\mathsf{true}} \cup (N(v) \cap C_i)$.*

Proof. Since $1 < |N(v) \cap C_i| \leq |C_i| - 1$, v has at least two neighbors $u, w \in C_i$ and at least one nonneighbor $x \in C_i$. If a vertex $z \in N(v) \cap C_i$ is in B, for any valid monopolar partition (A, B) of $V(G)$, then since B is an independent set, it follows that $C_i - \{z\} \subseteq A$. In particular, v is in A, at least one of u, w, say u, is in A, and x is in A. But this implies that (v, u, x) forms an induced P_3 in A, contradicting that A is a cluster graph. ◀

▶ **Reduction Rule 2.8.** (♠) *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$, and let C_i, C_j, $i < j$, be two cliques such that C_i is a large clique and C_j is either a large clique or an edge clique. If there are at least two edges between C_i and C_j then one of the following reductions, considered in the listed order, is applicable:*

Case (1) *There are two edges uu' and vv', where $u, v \in C_i$ and $u', v' \in C_j$, such that $u \neq v$ and $u' \neq v'$. Let $w \in C_i$ be such that $w \notin \{u, v\}$ (note that w exists because $|C_i| \geq 3$). Set $A_{\mathsf{true}} = A_{\mathsf{true}} \cup \{w\}$.*

Case (2) *$N(C_j) \cap C_i = \{v\}$. Set $B_{\mathsf{true}} = B_{\mathsf{true}} \cup \{v\}$.*

We can now bound the number of large cliques and edge cliques in yes-instances.

▶ **Reduction Rule 2.9.** (♠) *Let (G, k) be an instance of* Monopolar Recognition*, and let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. If in $(C_1, \ldots, C_r)$ either the number of large cliques is more than k, or the number of large cliques plus the number of edge cliques is more than $2k$, then reject the instance (G, k).*

▶ **Reduction Rule 2.10.** (♠) *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$, let C be a cluster in A_{true}, and let C_i, $i \in [r]$, be a large clique. If $v \in C_i$ is such that: (1) v is the only vertex in C_i that is adjacent to C, or (2) v is the only vertex in C_i that is not adjacent to C, then set $B_{\mathsf{true}} = B_{\mathsf{true}} \cup \{v\}$.*

▶ **Reduction Rule 2.11.** (♠) *Let (G, k) be an instance of* Monopolar Recognition*, and let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. If either (1) B_{true} contains more than $k + 1$ vertices or (2) there exists a cluster in A_{true} that is not a singleton, then reduce the instance (G, k) to an instance (G', k) with G' constructed as follows. Let $V(G') = V_1 \cup V_2 \cup V_3$, where $V_1 = \{u_C \mid C \text{ is a cluster in } A_{\mathsf{true}}\}$, $V_2 = \{v_1, \ldots, v_{k+1}\}$, and $V_3 = C_1 \cup \cdots \cup C_r$; and $E(G') = \{vu_C \mid v \in V_2 \wedge u_C \in V_1\} \cup \{vu_C \mid v \in V_3 \wedge u_C \in$*

$V_1 \wedge v$ is adjacent to $C\}$. That is, G' is constructed from G by introducing $k+1$ new vertices, replacing each cluster C in A_{true} (if any) by a single vertex u_C whose neighborhood is the neighborhood of C in $C_1, \ldots, C_r$ plus the $k+1$ new vertices, and keeping $C_1, \ldots, C_r$ the same.

If Reduction Rule 2.11 is applied, then after its application, we set A_{true} to V_1 and B_{true} to $\{v_1, \ldots, v_{k+1}\}$. Note that in any valid monopolar partition (A, B) of the graph resulting from the application of Reduction Rule 2.11, each vertex in V_1 must be in A, being adjacent to the $k+1$ independent set vertices $v_1, \ldots, v_{k+1}$, whereas the vertices $v_1, \ldots, v_{k+1}$ can be safely assumed to be in B since their only neighbors are in $V_1 \subseteq A$.

Modeling the Pushing Process by a Bipartite Graph. We now have bounded the number of large and edge cliques, and the size of A_{true} and B_{true}. It remains to bound the size of the large cliques and the number of vertex cliques. The challenge here is that we need to identify vertices such that putting them in A or B will eventually, after a series of pushes, lead either to the creation of too many clusters in A, or to the addition of two adjacent vertices in B. To model the avalanche of pushes to A or B, we introduce the following auxiliary graph.

▶ **Definition 2.12.** For a normalized partition $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ of $V(G)$, we define the auxiliary bipartite graph Λ as follows. The vertex set of Λ is $V(\Lambda) = V_C \cup V_I$, where V_C is the set of all vertices in the large cliques in $C_1, \ldots, C_r$, and V_I is the set of all vertices in the vertex cliques in $C_1, \ldots, C_r$. The edge set of Λ is $E(\Lambda) = \{uv \in E(G) \mid u \in V_C \text{ and } v \in V_I\}$; that is, $E(\Lambda)$ consists of precisely the edges in $E(G)$ that are between V_C and V_I.

Recall that V_I is an independent set in G by Fact 2.1. For a vertex $v \in V(\Lambda)$, we write $N_\Lambda(v)$ for the set of neighbors of v in Λ. We have the following lemma:

▶ **Lemma 2.13.** *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$ and consider the graph $\Lambda = (V(\Lambda), E(\Lambda))$. Then the maximum degree of Λ, $\Delta(\Lambda)$, is at most k.*

Proof. For every vertex $v \in V_C$, we have $|N_\Lambda(v)| \leq k$ because Reduction Rule 2.6 is inapplicable. By property (ii) of a nice decomposition and the inapplicability of Reduction Rule 2.7, every vertex clique that is adjacent to a large clique C is adjacent to exactly one vertex in C. Since by Reduction Rule 2.9 the number of large cliques is at most k, every vertex in V_I, which is a vertex clique by definition of V_I, has at most k neighbors in V_C. Therefore, for every vertex $v \in V_I$, we have $|N_\Lambda(v)| \leq k$. ◀

For two vertices $u, v \in V(\Lambda)$, write $dist_\Lambda(u, v)$ for the length of a shortest path between u and v in Λ. For a vertex $v \in V(\Lambda)$ and $i \in \{0, \ldots, n\}$, define $N^i(v) = \{u \in V(\Lambda) \mid dist_\Lambda(u, v) = i\}$. Write $\bar{0}_n$ (resp. $\bar{1}_n$) for the set of even (resp. odd) integers in $\{0, \ldots, n\}$.

▶ **Lemma 2.14.** *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$, let $\Lambda = (V(\Lambda), E(\Lambda))$ be the associated auxiliary graph, and let (A, B) be any valid monopolar partition of G.*

- **(i)** *For each $v \in V_C$: If $v \in B$ then $N_\Lambda(v) \subseteq A$.*
- **(ii)** *For each $v \in V_I$: If $v \in A$ then $N_\Lambda(v) \subseteq B$.*
- **(iii)** *For each $v \in V_C$: If $v \in B$ then $N^i_\Lambda(v) \subseteq B$ for $i \in \bar{0}_n$, and $N^i_\Lambda(v) \subseteq A$ for $i \in \bar{1}_n$.*
- **(iv)** *For each $v \in V_I$: If $v \in A$ then $N^i_\Lambda(v) \subseteq A$ for $i \in \bar{0}_n$, and $N^i_\Lambda(v) \subseteq B$ for $i \in \bar{1}_n$.*

Proof. (i): This trivially follows because B is an independent set.

(ii): Suppose that $v \in V_I$ is in A, and let $u \in N_\Lambda(v)$. Then $u \in V_C$ because Λ is bipartite, and hence, by definition, u belongs to a large clique C_i for some $i \in [r]$. Suppose, to get a contradiction, that $u \in A$. Since C_i is a large clique, and hence $|C_i| \geq 3$, there exists a

vertex $w \neq u$ in C_i such that $w \in A$. By property (ii) of the nice decomposition $(C_1, \ldots, C_r)$ and the inapplicability of Reduction Rule 2.7, $\{v, w\} \notin E(G)$. But this implies that (v, u, w) is an induced P_3 in A, contradicting that A is a cluster graph. It follows that $N_\Lambda(v) \subseteq B$.

(iii): This follows by repeated alternating applications of (i) and (ii) above.

(iv): This follows by repeated alternating applications of (ii) and (i) above. ◀

▶ **Reduction Rule 2.15.** (♠) *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$, and let $\Lambda = (V(\Lambda), E(\Lambda))$ be the associated auxiliary graph.*

- **(i)** *For any vertex $v \in V_C$: If either $\bigcup_{i \in \bar{0}_n} N^i_\Lambda(v)$ contains two adjacent (in G) vertices or $|\bigcup_{i \in \bar{1}_n} N^i_\Lambda(v)| > k$, then set $A_{\mathsf{true}} = A_{\mathsf{true}} \cup \{v\}$.*
- **(ii)** *For any vertex $v \in V_I$: If either $|\bigcup_{i \in \bar{0}_n} N^i_\Lambda(v)| > k$ or $\bigcup_{i \in \bar{1}_n} N^i_\Lambda(v)$ contains two adjacent (in G) vertices, then set $B_{\mathsf{true}} = B_{\mathsf{true}} \cup \{v\}$.*

Proof. (i) Let $v \in V_C$, and suppose that either $\bigcup_{i \in \bar{0}_n} N^i_\Lambda(v)$ contains two adjacent vertices or $|\bigcup_{i \in \bar{1}_n} N^i_\Lambda(v)| > k$. If $v \in B$ for any valid partition (A, B) of G, then by part (iii) of Lemma 2.14, it would follow that $\bigcup_{i \in \bar{0}_n} N^i_\Lambda(v) \subseteq B$ and $\bigcup_{i \in \bar{1}_n} N^i_\Lambda(v) \subseteq A$. In either case this contradicts that (A, B) is valid partition of G: If $\bigcup_{i \in \bar{0}_n} N^i_\Lambda(v)$ contains two adjacent vertices, then B is not an independent set, and if $|\bigcup_{i \in \bar{1}_n} N^i_\Lambda(v)| > k$ then A contains more than k clusters since $\bigcup_{i \in \bar{1}_n} N^i_\Lambda(v)$ induces an independent set in G.

(ii) The proof follows along the same lines as the proof of (i) (♠). ◀

▶ **Definition 2.16.** Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$, and let $\Lambda = (V(\Lambda), E(\Lambda))$ be the associated auxiliary graph. From each large clique C_i, $i \in [r]$, fix three vertices u_i, v_i, w_i; define $V_{\mathsf{fixed}} = \{u_i, v_i, w_i \mid C_i \text{ is a large clique}\}$ to be the set of all fixed vertices. Define $V_{\mathsf{edge}} = \{u \mid u \text{ is contained in some edge clique } C_i\}$ to be the set of vertices of the edge cliques, define $N_{\mathsf{edge}} = N(V_{\mathsf{edge}}) \cap V(\Lambda)$ to be the neighbors of V_{edge} in $V(\Lambda)$, and define $N^{\cup}_{\mathsf{edge}} = \bigcup_{v \in N_{\mathsf{edge}}} \bigcup_{i \leq n} N^i_\Lambda(v)$ to be the set of all vertices in $V(\Lambda)$ that are reachable in Λ from the vertices in N_{edge}. Define $V_{\mathsf{inter}} = \{u, v \mid u \in C_i \wedge v \in C_j \wedge i \neq j \wedge uv \in E(G) \wedge (C_i, C_j \text{ are large cliques})\}$ to be the set of endpoints of edges between large cliques, and define $N^{\cup}_{\mathsf{inter}} = \bigcup_{v \in V_{\mathsf{inter}}} \bigcup_{i \leq n} N^i_\Lambda(v)$ to be the set of all vertices in $V(\Lambda)$ that are reachable in Λ from the vertices in V_{inter}. Finally, let $V_{\mathsf{rep}} = A_{\mathsf{true}} \cup B_{\mathsf{true}} \cup V_{\mathsf{fixed}} \cup N^{\cup}_{\mathsf{inter}} \cup V_{\mathsf{edge}} \cup N^{\cup}_{\mathsf{edge}}$.

▶ **Reduction Rule 2.17.** (♠) *Let (G, k) be an instance of* Monopolar Recognition*, and let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. Let V_{rep} be as defined in Definition 2.16. Set $G = G[V_{\mathsf{rep}}]$.*

We now give the polynomial kernel whose existence was promised in Theorem 1.1.

▶ **Theorem 2.18.** Monopolar Recognition *has a kernel of size at most $9k^4 + 9k + 1$ which can be computed in $\mathcal{O}(n^2 m)$ time.*

Proof. Given an instance (G, k) of Monopolar Recognition, we apply Reduction Rules 2.3–2.17 exhaustively to (G, k). Clearly, the above rules can be applied in polynomial time. Let (G', k') be the resulting instance, let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G')$ with respect to which none of Reduction Rules 2.3–2.17 applies, and let $\Lambda = (V(\Lambda), E(\Lambda))$ be the auxiliary graph. Note that, by Reduction Rule 2.17, $V(G') = V_{\mathsf{rep}} = A_{\mathsf{true}} \cup B_{\mathsf{true}} \cup V_{\mathsf{fixed}} \cup N^{\cup}_{\mathsf{inter}} \cup V_{\mathsf{edge}} \cup N^{\cup}_{\mathsf{edge}}$. By Reduction Rule 2.9, the number of large cliques is at most k, and the number of edge cliques is at most $2k$. It follows that $|V_{\mathsf{fixed}}| \leq 3k$ and $|V_{\mathsf{edge}}| \leq 4k$. For a vertex $v \in V_{\mathsf{edge}}$, by Reduction Rule 2.6, v has at most k neighbors in V_I. Moreover, by Reduction Rule 2.8, v can have at most k neighbors in V_C, and therefore, $|N_\Lambda(v)| \leq 2k$, and $|N_{\mathsf{edge}}| \leq 4k \cdot 2k = 8k^2$. Since Reduction Rule 2.15 does not apply and $\Delta(\Lambda) \leq k$ by Lemma 2.13, we have that, for any $v \in V(\Lambda)$, we have

$|\bigcup_{i\leq n} N^i_\Lambda(v)| \leq \Delta(\Lambda)\cdot k \leq k^2$. This implies that $|N^{\cup}_{\mathsf{edge}}| \leq 8k^2\cdot k^2 \leq 8k^4$. Now since the number of large cliques is at most k, by Reduction Rule 2.8, it follows that $|V_{\mathsf{inter}}| \leq \binom{k}{2} < k^2$. Since for a vertex $v \in V(\Lambda)$ we have $|\bigcup_{i\leq n} N^i_\Lambda(v)| \leq k^2$ as argued above, it follows that $|N^{\cup}_{\mathsf{inter}}| \leq k^4$. Since $|A_{\mathsf{true}}| \leq k$ and $|B_{\mathsf{true}}| \leq k+1$, putting everything together, we conclude that the number of vertices in $V(G')$, $|V_{\mathsf{rep}}|$, is at most $k+k+1+3k+k^4+4k+8k^4 \leq 9k^4+9k+1$. The running time proof is omitted (♠). ◀

3 Kernel-size lower bound

This section is dedicated to proving the "only if" direction of Theorem 1.1, which, together with Theorem 2.18, completes its proof. In particular, we prove the following:

▶ **Theorem 3.1.** *Let* Π *be a graph property characterized by a (not necessarily finite) set* $\mathcal{H}$ *of connected forbidden induced subgraphs, each of order at least* 3*. Then unless* NP $\subseteq$ coNP/poly, Cluster-Π-Partition *parameterized by the number* k *of clusters in the cluster graph* $G[A]$ *does not admit a polynomial kernel.*

Throughout, let Π be any graph property satisfying the conditions of Theorem 3.1. We show Theorem 3.1 by giving a cross-composition from the NP-hard problem Colorful Independent Set [14]. Herein, we are given a graph $G=(V,E)$, $k\in\mathbb{N}$, and a proper k-coloring $c\colon V\to\{1,\ldots,k\}$; the question is whether there is an independent set with k vertices in G that contains exactly one vertex of each color. In the remainder of this section, we explain the construction behind the cross-composition and prove its correctness. We start by describing the intuition behind the construction, and why the avalanches in this case cannot be contained.

In contrast to Monopolar Recognition, the avalanches caused by the pushing process for the general Cluster-Π-Partition problem are much more uncontrollable: If some push to the Π-side B creates a forbidden induced subgraph M for Π in $G[B]$, we can repair the partition and "break" M by moving any vertex of M to the cluster graph side A. However, each move of a vertex in M may lead – through further necessary pushes from A to B – to distinct forbidden induced subgraphs in $G[B]$, again with multiple possible ways of breaking them in order to repair the partition. These avalanches cannot be contained, and lead to many possible paths along which they can be repaired, which can be modeled using a tree-like structure.

It is precisely the above-described behavior of avalanches that we exploit to obtain a cross-composition: The main gadgets select a Colorful Independent Set instance and independent-set vertices within that instance. Each such selection gadget has a trivial cluster-Π partition with one caveat: It has one (singleton) cluster too many in $G[A]$, and only this vertex can be pushed into the Π-side B. We call this vertex the *activator* vertex of the gadget. Pushing the activator vertex into B creates a forbidden induced subgraph for Π, requiring further pushes that propagate along a root-leaf path in a binary-tree-like structure. In the end, exactly one vertex corresponding to a leaf in this structure will be pushed from A to B, transmitting the choice to further gadgets.

Setup. Let t instances of Colorful Independent Set be given, with graphs $G_1,\ldots,G_t$, respectively. Below, we use an instance and its index in $[t]$ interchangeably. Without loss of generality, assume that the following properties hold; they can be achieved by simple padding techniques. Each instance asks for an independent set of size k, each color class in each graph has n vertices and n as well as t are powers of two. In the following, let m be the maximum number of edges over all graphs G_i.

We construct an instance of Cluster-Π-Partition as described below. The instance consists of the graph G and asks for a cluster-Π partition (A, B) with at most d clusters in $G[A]$ (we specify d below). The graph G is constructed by first adding d vertices which we call *anchors* (see below). The clusters in any cluster-Π partition (A, B) of G with d clusters in $G[A]$ will extend these anchor vertices into larger cliques. We then successively add gadgets that are attached to these anchors. We first construct an instance-selection gadget that selects one of the given t instances. Then we add a vertex-selection gadget for each instance which selects k vertices in its corresponding instance if it has been selected. Finally, we add verification gadgets that ensure that the selected vertices are pairwise nonadjacent in the graph of the selected instance.

Throughout, we use the following notation. We denote by (A, B) an arbitrary fixed cluster-Π partition of G. We fix M to be a forbidden induced subgraph of Π with minimum number of vertices. By assumption, M contains at least three vertices. The vertices that we introduce will be in three disjoint categories: *helper* vertices, *dial* vertices, and *volatile* vertices. Their meaning is as follows. Helper vertices will always be contained in B and only serve to impose certain properties on other vertices. Dial vertices are normally in A and belong to a cluster extending around an anchor; some of these vertices may be pushed to B by an avalanche. On the other hand, volatile vertices are normally in B and may be pushed to A by an avalanche.

First, we introduce d anchor vertices, divided into $5 + 2k$ groups: a_1^1, a_2^1; $a_1^2, \ldots, a_{2\log t}^2$; $a_1^3, \ldots, a_{k+1}^3$; for each $i \in [k]$, $a_1^{3+i}, \ldots, a_{\log n}^{3+i}$; for each $i \in [k]$, $a_1^{3+k+i}, \ldots, a_n^{3+k+i}$; and $a_1^{5+2k}, \ldots, a_m^{5+2k}$. Hence, we put $d \coloneqq 2 + 2\log(t) + k + k\log n + kn + 2m$. The groups of anchors correspond to the gadgets constructed below in which they are used. Each anchor vertex is a dial vertex. We fix each of the anchors into A by introducing, for each anchor a_i^j, $d + 1$ copies of M and, for each copy, identifying an arbitrary vertex of that copy with a_i^j. The vertices different from a_i^j in the copies of M are helper vertices. If $a_i^j \in B$, then out of each of the d incident copies of M, at least one vertex is in A, and since these vertices are pairwise nonadjacent, $G[A]$ would contain at least $d + 1$ clusters, which is a contradiction. Thus, each anchor must be in A. When we construct cluster-Π partitions in the following we always tacitly assume that anchors are in A and all helper vertices are in B.

We associate each anchor a_i^j with a vertex set D_i^j that contains a_i^j and induces a clique in G (throughout the construction). We say that D_i^j is the *dial* of a_i^j. Initially, $D_i^j = \{a_i^j\}$. Later on, other vertices may *join* D_i^j; by saying a vertex v *joins* D_i^j, we mean that we put v into D_i^j and make v adjacent to all other vertices in D_i^j. Intuitively, the set of anchors corresponds to the clusters in $G[A]$. These clusters are divided into two types: Either an anchor's dial contains at least two vertices and the cluster consists only of vertices in the anchor's dial, or the anchor's dial contains only the anchor, and a single volatile vertex may join the anchor's cluster. We use the following notation.

▶ **Definition 3.2.** Let (A, B) be a cluster-Π partition for G and $\mathcal{D}$ be a set of dials. Partition (A, B) is *friendly* with respect to $\mathcal{D}$ if each singleton dial in $\mathcal{D}$ is a singleton cluster in $G[A]$.

Next, we introduce the operation of making three vertices exclusive. Intuitively, this operation is our main tool to fan out the possible pushes in avalanches according to a binary tree: When u is pushed to B, either v or w can be pushed to A to repair the partition. We use this construction extensively in the selection gadgets described below.

Given three vertices $u, v, w \in V(G)$, by *making u, v, and w exclusive* we mean: (i) introducing a copy of M into G, (ii) identifying three distinct vertices of M with u, v, and w, respectively, and (iii) fixing all remaining vertices of M (if any) into B by making each of them adjacent to both a_1^1 and a_2^1. The vertices in $V(M) \setminus \{u, v, w\}$ are helper vertices. Observe that $V(M) \setminus \{u, v, w\} \subseteq B$, because, otherwise, there would be a P_3 in $G[A]$ involving a_1^1 and a_2^1. Furthermore, not all three $u, v, w \in B$ since otherwise $G[B]$ contains a copy of M. When constructing cluster-Π partitions we will always tacitly assume that $V(M) \setminus \{u, v, w\} \subseteq B$ and ignore the vertices in $V(M) \setminus \{u, v, w\}$. Furthermore, to simplify showing that the constructed partition (A, B) is a cluster-Π partition we will show that $G[A]$ is a cluster graph, that $G[B] - \{u, v, w\} \in \Pi$, that at least one of u, v, w is in A and that $\{u, v, w\} \cap B$ do not have any neighbors in $G[B]$ other than $\{u, v, w\}$. Since Π is characterized by connected forbidden induced subgraphs and the helper vertices will not receive further neighbors, this suffices to prove that $G[B] \in \Pi$.

Instance Selection. The inner workings of the generic selection gadget described below use the necessary pushes along a binary-tree-like structure outlined above.

For use as an instance-selection gadget, we need to take special care so that the number of clusters used is roughly logarithmic in the number of instances. We achieve this by using only two clusters (represented by anchors and their dials) per level in the binary-tree-like structure of pushes. For use as a vertex-selection gadget, to bound the number of clusters in the size of the largest instance, we need to ensure that all the vertex-selection gadgets share their corresponding clusters. We achieve this by grouping the gadgets according to the groups of anchors above; each gadget uses only anchors in their corresponding group and shares these anchors with all other gadgets in this group. Essentially, the operation of vertices joining dials makes it possible to define the selection gadgets in a relatively local way.

We will use the following (generic) construction both for selecting an instance and for selecting the independent-set vertices in that instance. For this purpose, fix two construction parameters $p, q \in \mathbb{N}$, where p specifies which anchors (and dials) we use when constructing the gadget and q specifies how many possible choices shall be modeled. Herein, we require that q be a power of two. For the instance-selection gadget we will set $p = 2$ and $q = t$.

We introduce a new vertex v^*. Our goal is to construct a structure in which, starting from a trivial cluster-Π partition (A, B), putting $v^* \in B$ triggers an avalanche of pushes according to a path in a binary-tree-like structure. To this end, fix a rooted binary tree T with q leaves (corresponding to the $q = t$ instances of Colorful Independent Set for the instance-selection gadget). Say a vertex in T is on *level* $i \in [\log q]$ if its distance from the root is i. For $i \in [\log q]$, L_i denotes the set of vertices at level i. The tree T will not be part of the constructed graph, we use it only as a scaffold to define the actual vertices in the graph.

For each vertex $v \in V(T)$ except the root, introduce two vertices $\alpha(v), \beta(v)$ into G. Let i be the level of v. Connect $\alpha(v)$ to both a_{2i-1}^p and $\beta(v)$. Make $\beta(v)$ join D_{2i}^p. Furthermore, for each vertex $u \in L_i$, $i \in \{0, \ldots, \log q\}$, let v, w be the two children of u in T and make $\beta(u), \alpha(v), \alpha(w)$ exclusive. If $i = 0$, then let v, w be the two vertices in level 1 in T and make $v^*, \alpha(v), \alpha(w)$ exclusive instead. This completes the construction of the selection gadget. Vertex v^* is a volatile vertex, as is $\alpha(v)$ for $v \in V(T)$. Each $\beta(v)$, $v \in V(T)$, is a dial vertex. Call the constructed gadget $\mathsf{selection}(p, q)$, and say that v^* is the *activator vertex*, and that the vertices in $\{\beta(v) \mid v \in L_{\log q}\}$ are the *choice vertices*. We fix an arbitrary order of the choice vertices, so that we may speak of the ith choice vertex without confusion.

▶ **Lemma 3.3.** (♠) *Let G' be the graph before applying* selection(p, q) *and G the graph afterwards.*

- **(i)** *If cluster-Π partition (A, B) has at most d clusters in $G[A]$ and the activator vertex is in B, then at least one choice vertex is in B.*
- **(ii)** *If there is a cluster-Π partition (A', B') for G' with d clusters in $G'[A']$, then there is a cluster-Π partition (A, B) for G with $d+1$ clusters, where the activator vertex is a singleton cluster and each choice vertex is in A. If (A', B') is friendly with respect to the dials D_i^p, then (A, B) is friendly with respect to the dials D_i^p.*
- **(iii)** *If G' has a cluster-Π partition (A', B') that is friendly with respect to the dials D_i^p and such that $G'[A']$ contains at most d clusters, then, for each $i \in [q]$, there is a cluster-Π partition (A, B) of G, such that graph $G[A]$ contains at most d clusters, and out of all choice vertices only the ith one is in B (and, necessarily, the activator vertex is in B). Moreover, the choice vertex that is contained in B is isolated in $G[B]$.*

As mentioned, to construct the *instance-selection gadget*, we carry out selection$(2, t)$. For further reference, fix a bijection ϕ from the set of instances $[t]$ to the choice vertices produced by the construction. We use ϕ later to denote the choice vertex corresponding to an instance.

Vertex Selection. We now use the above construction selection$(\cdot, \cdot)$ to create vertex-selection gadgets for each instance and each color. Each vertex-selection gadget selects one vertex of the gadget's color into an independent set when activated by putting the activator vertex into B (which will be effected by the instance-selection gadget). The vertex-selection gadgets for each instance are distinct, but they use dials which are shared by all instances.

In the first part of the construction of the vertex-selection gadgets, for each instance $r \in [t]$ and color $i \in [k]$, carry out selection$(3 + i, n)$. Let $\psi^*_{r,i}$ be the corresponding activator vertex and fix a bijection $\psi_{r,i}$ from the vertices $V(G_r)$ of color i to the choice vertices. Make $\psi^*_{r,i}$ join D^3_{1+i}. Intuitively, if the activator vertex $\psi^*_{r,i}$ is put into B, the subgraph constructed by selection$(3 + i, n)$ enforces the push of a choice vertex into B, which by bijection $\psi_{r,i}$ correspond one-to-one to the vertices of color i in instance r. In this way, we model the selection of an independent-set vertex.

In the second part of the construction of the vertex-selection gadgets, we introduce a way to activate the vertex-selection gadgets of all colors if some instance $r \in [t]$ has been chosen. For this, carry out the following steps for each $r \in [t]$. Introduce two vertices u_r, v_r. Make $\phi(r)$, u_r, and v_r exclusive. Fix $u_r \in B$ by making it adjacent to both a_1, a_2. Make v_r adjacent to a^3_1 and, for each $i \in [k]$, make v_r adjacent to $\psi^*_{r,i}$. Vertex u_r is a helper vertex and v_r is a volatile vertex. This concludes the construction of the vertex-selection gadgets.

Intuitively, the selection of instance r is indicated by the fact that $\phi(r) \in B$. Since $u_r \in B$ and $\phi(r)$, u_r, and v_r are exclusive, $v_r \in A$. Vertex v_r forms a P_3 with a^3_1 and each $\psi^*_{r,i}$. Hence, the activator vertices $\psi^*_{r,i}$ of each vertex-selection gadget for instance r are in B. This enforces the selection of an independent-set vertex of each color.

By iteratively applying Lemma 3.3, we can show that the above-constructed graph has the properties that, if there is a cluster-Π partition (A, B) with d clusters in $G[A]$, then there is an instance for which the vertex-selection gadget for each color has one choice vertex in B (that is, the corresponding vertex is selected); and, vice-versa, for each possible selection of one vertex of each color in an instance, there is a corresponding cluster-Π partition.

Verification. For the verification gadgets it is again crucial to share clusters (anchors) between many gadgets to keep the overall number of clusters in A small. For this, we use $|V| = k \cdot n$ anchors that each represents, for each instance, one fixed vertex, and m pairs of anchors that each represents, for each instance, one fixed edge.

Due to space constraints, the details of the construction are not given here, but the working principle is as follows. Selecting a vertex v via a vertex-selection gadget will make it necessary to push a vertex corresponding to v into the cluster of its associated anchor. This push creates a P_3 in A for each incident edge e, necessitating a further push. Namely, we are required to push a vertex out of the cluster in A corresponding to one anchor associated with e. Pushing the corresponding vertex for the other endpoint of e into B will complete a forbidden induced subgraph, yielding that no two endpoints of an edge are selected. For the other direction of the correctness proof, we show that it is possible to configure the gadgets accordingly if one of the input instances is positive, which then concludes the proof of Theorem 3.1.

References

1 Faisal N. Abu-Khzam, Carl Feghali, and Haiko Müller. Partitioning a graph into disjoint cliques and a triangle-free graph. *Discrete Appl. Math.*, 190-191:1–12, 2015.

2 Demetrios Achlioptas. The complexity of G-free colourability. *Discrete Math.*, 165–166(0):21–30, 1997.

3 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.

4 Marin Bougeret and Pascal Ochem. The complexity of partitioning into disjoint cliques and a triangle-free graph. *Discrete Appl. Math.*, 217:438–445, 2017.

5 Hajo Broersma, Fedor V. Fomin, Jaroslav Nešetřil, and Gerhard J. Woeginger. More about subcolorings. *Computing*, 69(3):187–203, 2002.

6 Sharon Bruckner, Falk Hüffner, and Christian Komusiewicz. A graph modification approach for finding core-periphery structures in protein interaction networks. *Algorithms Mol. Biol.*, 10:16, 2015.

7 Zh. A. Chernyak and A. A. Chernyak. About recognizing (α, β) classes of polar graphs. *Discrete Math.*, 62(2):133–138, 1986.

8 Ross Churchley and Jing Huang. On the polarity and monopolarity of graphs. *J. Graph Theory*, 76(2):138–148, 2014.

9 Ross Churchley and Jing Huang. Solving partition problems with colour-bipartitions. *Graph. Combinator.*, 30(2):353–364, 2014.

10 Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

11 Reinhard Diestel. *Graph Theory, 4th Edition*. Springer, 2012.

12 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, Berlin, Heidelberg, 2013.

13 Alastair Farrugia. Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard. *Electron. J. Comb.*, 11(1):R46, 2004.

14 Michael R. Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.

15 Jirí Fiala, Klaus Jansen, Van Bang Le, and Eike Seidel. Graph subcolorings: Complexity and algorithms. *SIAM J. Discrete Math.*, 16(4):635–650, 2003.

16 Stéphane Foldes and Peter L. Hammer. Split graphs. *Congr. Numer.*, 19:311–315, 1977.

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 John Gimbel and Chris Hartman. Subcolorings and the subchromatic number of a graph. *Discrete Math.*, 272:139–154, 2003.

19 Pinar Heggernes, Dieter Kratsch, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Fixed-parameter algorithms for Cochromatic Number and Disjoint Rectangle Stabbing via iterative localization. *Infor. Comput.*, 231:109–116, 2013.

20 Iyad Kanj, Christian Komusiewicz, Manuel Sorge, and Erik Jan van Leeuwen. Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs. *J. Comput. Syst. Sci.*, 92:22–47, 2018.

21 Sudeshna Kolay, Fahad Panolan, Venkatesh Raman, and Saket Saurabh. Parameterized Algorithms on Perfect Graphs for Deletion to (r, l)-Graphs. In *Proc. 41st MFCS*, volume 58 of *LIPIcs*, pages 75:1–75:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

22 Jan Kratochvíl and Ingo Schiermeyer. On the computational complexity of $(\mathcal{O}, \mathcal{P})$-partition problems. *Discuss. Math. Graph Theory*, 17(2):253–258, 1997.

23 Van Bang Le and Ragnar Nevries. Complexity and algorithms for recognizing polar and monopolar graphs. *Theor. Comput. Sci.*, 528:1–11, 2014.

24 Juraj Stacho. On 2-subcolourings of chordal graphs. In *Proc. 8th LATIN*, volume 4957 of *LNCS*, pages 544–554. Springer, 2008.