SIAM J. DISCRETE MATH.
Vol. 34, No. 1, pp. 640–681

SOLVING PARTITION PROBLEMS ALMOST ALWAYS REQUIRES PUSHING MANY VERTICES AROUND*

IYAD KANJ†, CHRISTIAN KOMUSIEWICZ‡, MANUEL SORGE§, AND ERIK JAN VAN LEEUWEN¶

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. In this paper, we study whether certain fixed-parameter tractable (Π_A, Π_B)-RECOGNITION problems admit polynomial 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 $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, (Π_A, Π_B)-RECOGNITION admits a polynomial kernel if and only if $\mathcal{H}$ contains a graph with at most two vertices. In both the kernelization and the lower bound results, we exploit the properties of a *pushing process*, which is an algorithmic technique used recently by Heggerness et al. and by Kanj et al. to obtain fixed-parameter algorithms for many cases of (Π_A, Π_B)-RECOGNITION, as well as several other problems.

Key words. polynomial kernel, graph partitioning, monopolar graphs

AMS subject classifications. 05C85, 68Q17, 68Q25, 68R10, 68W40

DOI. 10.1137/19M1239362

1. Introduction.

Given two (induced-)hereditary graph properties Π_A and Π_B, a graph G is a (Π_A, Π_B)-*graph* 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 generic problem 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.[1]

Note that, since Π_A and Π_B are hereditary, they are characterized by a (not

*Received by the editors January 22, 2019; accepted for publication (in revised form) January 6, 2020; published electronically March 10, 2020. This manuscript is the full version of an article in the *Proceedings of the 26th Annual European Symposium on Algorithms (ESA '18)* [24].
https://doi.org/10.1137/19M1239362
Funding: The second author gratefully acknowledges support by the DFG, project MAGZ, KO 3669/4-1. The third author 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 631163.11, by the Israel Science Foundation (grant 551145/14), and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement 714704.

†School of Computing, DePaul University, Chicago, IL 60604-2301 (ikanj@cs.depaul.edu).
‡Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, 35037 Marburg, Germany (komusiewicz@informatik.uni-marburg.de).
§Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer Sheva, Israel, and Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, 02-097 Warsaw, Poland (manuel.sorge@mimuw.edu.pl).
¶Department of Information and Computing Sciences, Utrecht University, 3584 CC Utrecht, The Netherlands (e.j.vanleeuwen@uu.nl).

[1]The order of a graph is its number of vertices.

necessarily finite) set of forbidden induced subgraphs. In the most interesting cases, the characterization of one of the two properties Π_A and Π_B includes only forbidden induced subgraphs whose order is at least three. Indeed, if the characterizations of Π_A and Π_B both include a forbidden induced subgraph of order exactly two, then (Π_A, Π_B)-Recognition can be solved in linear time: The property Π_A, and similarly Π_B, is either the class of complete graphs or the class of edgeless graphs. Thus, the set of (Π_A, Π_B)-graphs is either the set of split graphs, which can be recognized in linear time [18], the set of bipartite graphs, or the set of co-bipartite graphs, which both are also well known to be recognizable in linear time. In this manuscript we thus focus on the case where the forbidden induced subgraph characterization of Π_A or Π_B does not include graphs of size at most two.

When Π_A or Π_B is characterized by larger forbidden induced subgraphs, then (Π_A, Π_B)-Recognition is more computationally complex [2, 14, 26]. In particular, (Π_A, Π_B)-Recognition is NP-hard for all Π_A and Π_B that are characterized by a set of *connected* forbidden induced subgraphs as long as at least one of Π_A and Π_B has only forbidden induced subgraphs of order at least three [14]. The restriction to connected forbidden induced subgraphs in this statement is necessary in order to get such a sweeping classification result. If we allow disconnected forbidden induced subgraphs in the characterizations of Π_A or Π_B, then there are some polynomial-time solvable cases. Consider, for example, the set of *unipolar graphs* which can be recognized in polynomial time [13, 18, 29, 32]. Unipolar graphs are (Π_A, Π_B)-graphs wherein Π_A is the set of complete graphs, which is characterized by forbidding the graph containing two nonadjacent vertices, and Π_B is the set of cluster graphs, which is characterized by forbidding the (simple) path P_3 on three vertices. Indeed, the general NP-hardness result from above does not apply to unipolar graphs, because, while the forbidden subgraph characterization for Π_B does not contain a graph of order at most two, the set of complete graphs is characterized by a *disconnected* forbidden induced subgraph. There are further similar polynomial-time solvable cases of (Π_A, Π_B)-Recognition [23]. We are here concerned with the NP-hard cases of (Π_A, Π_B)-Recognition, and thus we focus on the cases where both Π_A and Π_B are characterized only by connected forbidden subgraphs. Note that, equivalently, Π_A and Π_B are each closed under the disjoint union of graphs.[2]

Many (Π_A, Π_B)-Recognition problems that can be characterized by forbidden induced subgraphs were shown to be fixed-parameter tractable (FPT)—for instance, when Π_A is the class of graphs that is a disjoint union of k cliques and Π_B is the set of edgeless graphs or the set of cluster graphs [23], where k is the parameter. We thus aim here to complement these results by studying the kernelization complexity of (Π_A, Π_B)-Recognition.

By the discussion above, the first interesting case of (Π_A, Π_B)-Recognition to study with respect to kernelization is when the two properties are closed under the disjoint union and one of the two properties, say Π_A, is characterized by connected forbidden induced subgraphs of size no less than three. We thus consider the first level above triviality of (Π_A, Π_B)-Recognition by letting Π_A be the hereditary class characterized by a single forbidden induced subgraph, which, in addition, is the most simple connected graph with at least three vertices: the path P_3. This leads to the following problem.

[2]That is, the disjoint union of two graphs that each satisfy Π_A (resp., Π_B) also satisfies Π_A (resp., Π_B).

CLUSTER-Π-PARTITION
Input: A graph $G = (V, E)$.
Question: Can $V(G)$ be partitioned into (A, B) 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, 8, 9, 28] (Π is the set of edgeless graphs), 2-*subcolorable* graphs [5, 16, 21, 31] (Π is the set of cluster graphs), and several others [1, 4, 7]. CLUSTER-Π-PARTITION is NP-hard in these special cases, and by the general hardness result mentioned above, it is NP-hard for all Π that are characterized by a set of connected forbidden induced subgraphs [2, 14, 26]. To cope with this hardness, we consider the number k of clusters in the cluster graph $G[A]$ as a parameter. As mentioned above, this parameter led to a number of tractability results for CLUSTER-Π-PARTITION. In particular, recognizing monopolar graphs and recognizing 2-subcolorable graphs are fixed-parameter tractable (FPT) with respect to this parameter [23].

Our results. The result we obtain gives a complete characterization of the kernelization complexity of CLUSTER-Π-PARTITION.

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* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, 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 two.*

The positive result in Theorem 1.1 corresponds to the recognition of monopolar graphs. Indeed, the graph properties with forbidden induced subgraphs of order two are "being edgeless" and "being nonedgeless," but the latter is not characterized by connected forbidden induced subgraphs. Moreover, one application of Theorem 1.1 is to show that the recognition of 2-subcolorable graphs does not admit a polynomial kernel parameterized by the number of clusters in $G[A]$.

One might also 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 [28]. Thus, we consider the number k of vertices in the graph $G[B]$, even for the broader (Π_A, Π_B)-RECOGNITION problem. We previously proved a general fixed-parameter tractability result in this case [23]. Here, we observe a very general kernelization result.

THEOREM 1.2. (Π_A, Π_B)-RECOGNITION *parameterized by k, the maximum size of B, admits a polynomial kernel with $\mathcal{O}((d + 1)!(k + 1)^d)$ vertices, 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 a better bound on the number of vertices in the kernel 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, admits a polynomial kernel with $\mathcal{O}((\Delta^2 + 1) \cdot k^2)$ vertices.*

Our techniques. We obtain our main kernelization result, stated in Theorem 1.1, by studying an algorithmic method that we call the *pushing process*. It was employed in [22, 23, 25] to obtain fixed-parameter tractability results for several problems. In the context of (Π_A, Π_B)-RECOGNITION, the pushing process was employed as part of an algorithmic technique, referred to as *inductive recognition*, that works as follows [23]: The algorithm for the (Π_A, Π_B)-RECOGNITION problem under consideration empties the input graph and adds vertices back one by one while maintaining a valid partition. However, adding a vertex might invalidate a previously valid partition. To remedy this, the pushing process comes in: vertices are *pushed* from one part of the partition to the other part in the hope of obtaining a valid partition again. Earlier, Heggernes et al. [22] had employed a pushing process, as part of an algorithmic technique, referred to as *iterative localization*, which works very similarly to inductive recognition, 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). Kolay et al. [25] also applied it in follow-up work related to the cochromatic number. All aforementioned results, for the various problems under consideration, rely on iteratively/inductively maintaining a valid partition of the input instance, as the instance elements (vertices, rectangles, etc.) are added one at a time. After each element is added, which may invalidate the partition, a pushing process is applied in order to try to repair the partition. This process results in pushing some elements from each part of the partition to the other part and may have a cascading effect.

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 [22, 25]. The contribution of the previous works [22, 23, 25] 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 vertices relevant to avalanches 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. [25] 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, each of which induces a clique, and ℓ sets, each of which induces an edgeless graph. This problem has a parameterized algorithm using a pushing process [22], but Kolay et al. [25] 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.

In this work, we study the role that the pushing process plays in characterizing the kernelization complexity of (Π_A, Π_B)-RECOGNITION. The pushing process, and a deeper understanding of the avalanches it causes, turn out to be central to both directions of Theorem 1.1. Indeed, we show that, while for a specific Π the pushing process can be used to witness a small vertex set of size $k^{\mathcal{O}(1)}$ containing the vertices affected by avalanches, for all other Π such a set of polynomial size is unlikely to exist.

To prove the positive result in the theorem, 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 represent the structure of the avalanches that occur using an auxiliary 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 three, 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 $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, by providing a cross-composition.

2. Preliminaries.

For $\ell \in \mathbb{N}$, we use $[\ell]$ to denote $\{1, 2, \ldots, \ell\}$.

Graphs. We follow standard graph-theoretic notation [11]. Let G be a graph. By $V(G)$ and $E(G)$ we denote the vertex-set and the edge-set of G, respectively. Throughout the paper, we use $n := |V(G)|$ to denote the number of vertices in G and $m := |E(G)|$ to denote its number of edges. We also say that G is of *order* $|V(G)|$. We assume $n = \mathcal{O}(m)$ since isolated vertices can be safely removed in the problems that we consider. For $X \subseteq V(G)$, $G[X] = (X, \{e \mid e \in E(G) \cap X\})$ denotes the *subgraph of G induced by X*. For a vertex $v \in G$, $N(v) = \{u \mid \{u,v\} \in E(G)\}$ and $N[v] = N(v) \cup \{v\}$ denote the *open neighborhood* and the *closed neighborhood* of v, respectively. For $X \subseteq V(G)$, we define $N(X) := (\bigcup_{v \in X} N(v)) \setminus X$ and $N[X] := \bigcup_{v \in X} N[v]$, and for a family $\mathcal{X}$ of subsets $X \subseteq V(G)$, we define $N(\mathcal{X}) := (\bigcup_{X \in \mathcal{X}} N(X)) \setminus (\bigcup_{X \in \mathcal{X}} X)$ and $N[\mathcal{X}] := \bigcup_{X \in \mathcal{X}} N[X]$.

We say that a vertex v is *adjacent to a subset $X \subseteq V(G)$ of vertices* if v is adjacent to at least one vertex in X. Similarly, we say that two vertex sets $X \subseteq V(G)$ and $Y \subseteq V(G)$ are adjacent if there exist $x \in X$ and $y \in Y$ that are adjacent. If X is any set of vertices in G, we write $G - X$ for $G - X$. For a vertex $v \in V(G)$, we write $G - v$ for $G - \{v\}$.

Graph partitions. 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. The Monopolar Recognition problem is defined as follows.

> Monopolar Recognition
> **Input:** A graph $G = (V, E)$ and an integer k.
> **Question:** Does G admit 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.

Parameterized complexity. A *parameterized problem* is a tuple (P, κ), where $P \subset \Sigma^*$ is a language over some finite alphabet Σ and $\kappa \colon \Sigma^* \to \mathbb{N}$ is a *parameterization*. For a given instance $x \in \Sigma^*$, we also say $\kappa(x)$ is the *parameter*. A parameterized problem (P, κ) is *fixed-parameter tractable* (FPT) if there exists an algorithm that on input $x \in \Sigma^*$ decides if x is a yes-instance of P, that is, $x \in P$, and that runs in time $f(\kappa(x)) n^{\mathcal{O}(1)}$, where f is a computable function independent of $n = |x|$. A parameterized problem is *kernelizable* if there exists a polynomial-time reduction that maps an instance x of the problem to another instance x' such that (1) $|x'| \leq \lambda(\kappa(x))$ for some computable function λ, (2) $\kappa(x') \leq \lambda(\kappa(k))$, and (3) x is a yes-instance of the problem if and only if x' is. The instance x' is called the *kernel* of x, and $|x'|$ is the *kernel size*. It is well known that a parameterized problem is FPT if and only if it is kernelizable [12, 19], and a natural question to ask for an FPT problem is whether or not it has a kernel of polynomial size. We refer the reader to [19] for an in-depth discussion about kernelization.

Let $Q \subseteq \Sigma^*$ be a language and (P, κ) a parameterized problem; i.e., P is a language and $\kappa\colon \Sigma^* \to \mathbb{N}$ is a parameterization. An *or-cross-composition* from Q into (P, κ) is a polynomial-time algorithm that, given t instances $q_1, \dots, q_t \in \Sigma^*$ 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 $i \in [t]$. We have the following.

THEOREM 2.1 (see [3]). *Let $Q \subseteq \Sigma^*$ be an NP-hard language and (P, κ) be a parameterized problem. If there is an or-cross-composition from Q into (P, κ) and (P, κ) admits a polynomial-size problem kernel, then* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

For more discussion on parameterized complexity, we refer the reader to the literature [10, 12].

3. 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 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 in bounding the size of the instance.

3.1. 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 $(C_1, \dots, C_r)$ 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 $C_1, \dots, C_r$, $r \geq 1$, such that the tuple $(C_1, \dots, C_r)$ satisfies the following properties (see Figure 1 for an illustration):

(i) In the decomposition tuple $(C_1, \dots, C_r)$, 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 $i < j$.

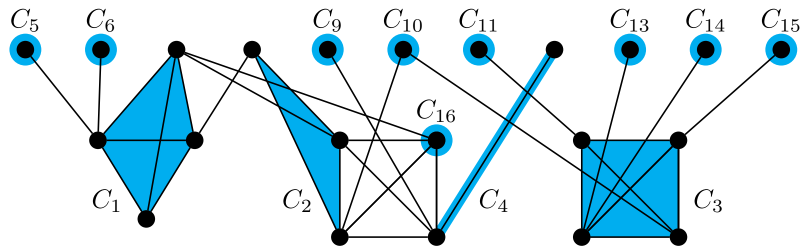

FIG. 1. *A monopolar graph with a clique decomposition. The large cliques are C_1, C_2, and C_3; the edge clique is C_4; all other cliques are small cliques. The small cliques form an independent set.*

(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 3.1. *The vertex cliques in a nice clique decomposition form an independent set in G.*

A nice clique decomposition of $V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$ can be computed as follows. Let $V' = V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}}) \neq \emptyset$. We check whether $G[V']$ contains a clique C of size at least three. If this is the case, then we find a maximal clique $C' \supseteq C$ in $G[V']$, add C' as a large clique to the decomposition, set $V' \leftarrow V' - C'$, and repeat. Otherwise, $G[V']$ does not contain any clique of size three, we check whether $G[V']$ contains an edge clique C (i.e., two endpoints of an edge), add C to the decomposition, set $V' \leftarrow V' - C$, and repeat. If no edge clique exists in $G[V']$, then the remaining vertices in V' form an independent set, and we add each one of them to the decomposition as a vertex clique. This process can be seen to run in polynomial time, but we will use the following more precise bound.

LEMMA 3.2. *A nice clique decomposition of G can be computed in $\mathcal{O}(nm)$ time.*

Proof. First, in $\mathcal{O}(nm)$ time, compute a list of *all* triangles in G. Then, label all vertices as *free*. Let G' denote the graph $G[V']$. Process the list from head to tail; that is, consider each triangle in the list. If one vertex of the triangle is not labeled as free, then continue with the next triangle. If all vertices in this triangle are labeled as free, then compute a maximal clique in G' containing this triangle and consisting only of free vertices. This can be done in $\mathcal{O}(m)$ time [30]. Add the maximal clique to the decomposition as described above, remove all vertices of the maximal clique from G', and unlabel all vertices of the maximal clique. Overall this step takes $\mathcal{O}(nm)$ time, since we encounter at most $n/3$ triangles whose vertices are labeled free. Once all triangles in the list are processed, compute a set of edge cliques in $\mathcal{O}(m)$ time by computing a maximal matching in $G[V']$. Finally, add all remaining vertices as vertex cliques in $\mathcal{O}(n)$ time. ∎

Let (G, k) be an instance of MONOPOLAR RECOGNITION. We initialize $A_{\mathsf{true}} = B_{\mathsf{true}} = \emptyset$ and $V' = V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$, 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 $V(G)$.

3.2. 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, with respect to the order in which the rules are listed, is applicable.

The following rule is straightforward.

REDUCTION RULE 3.3. *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 3.4. *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 $A_{\mathsf{true}} \subseteq A$ and the cluster graph $G[A]$ is P_3-free for every monopolar partition (A,B) of G.

REDUCTION RULE 3.5. *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1,\ldots,C_r)$ be a normalized partition of $V(G)$. If there is a vertex $v \in V(G)\setminus(A_{\mathsf{true}}\cup B_{\mathsf{true}})$ such that $G[A_{\mathsf{true}}\cup\{v\}]$ contains an induced P_3, then set $B_{\mathsf{true}} = B_{\mathsf{true}}\cup\{v\}$.*

After the exhaustive application of the above rules, we have that $G[A_{\mathsf{true}}]$ is a cluster graph, $G[B_{\mathsf{true}}]$ fulfills Π, and any vertex in $V(G)\setminus(A_{\mathsf{true}}\cup B_{\mathsf{true}})$ can be moved to either side of the partition without creating a forbidden induced subgraph.

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

REDUCTION RULE 3.6. *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 $1 < |N(v)\cap C_i| \le |C_i|-1$, then set $A_{\mathsf{true}} = A_{\mathsf{true}}\cup(N(v)\cap C_i)$.*

Correctness proof. Since $1 < |N(v)\cap C_i| \le |C_i|-1$, 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 3.7. *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 $u \ne v$ and $u' \ne v'$. Let $w \in C_i$ be such that $w \notin \{u,v\}$ (note that w exists because $|C_i| \ge 3$). Set $A_{\mathsf{true}} = A_{\mathsf{true}}\cup\{w\}$.*

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

Correctness proof. We first prove that either case (1) or case (2) applies. Suppose that case (1) does not apply, and we show that case (2) does.

By maximality of C_i in $\bigcup_{j\geq i} C_j$ (property (ii) in the definition of a nice clique decomposition), no vertex in C_j can be adjacent to all vertices in C_i. It follows from this fact and from the inapplicability of Reduction Rule 3.6 that each vertex in C_j has at most one neighbor in C_i. Since case (1) does not apply, the vertices in C_j that have a neighbor in C_i must all have the same neighbor, which proves that case (2) applies.

Now suppose that case (1) applies, and we will show the correctness of the reduction rule in this case. Let (A, B) be any valid monopolar partition of (G, k). Since at most one of u', v' can be in B, at least one of u', v', say u', is in A. Suppose, to get a contradiction, that $w \in B$. Then both u and v must be in A. By maximality of C_i in $\bigcup_{j\geq i} C_j$, u' cannot be adjacent to all vertices in C_i. Since Reduction Rule 3.6 is not applicable, u must be the only neighbor of u' in C_i. But then (v, u, u') is an induced P_3 in A, contradicting that A is a cluster graph.

Suppose that case (2) applies, and suppose to get a contradiction that $v \in A$ in some valid monopolar partition (A, B) of (G, k). Since there are at least two edges between C_i and C_j, v has at least two neighbors $u', v' \in C_j$. Again, observe that at least one of u', v', say v', must be in A. Since $|C_i| \geq 3$, at least one vertex in C_i, say w, must be in A. Since v is the only neighbor of v' in C_i by the premise of case (2), it follows that (w, v, v') is an induced P_3 in A, contradicting that A is a cluster graph. □

After applying Reduction Rules 3.6 and 3.7, it holds that no two large cliques and no large clique and an edge clique can merge into a single cluster. Hence, any large clique will create a new cluster in A, and edge cliques cannot join any large clique in A. Based on that, we can now upper bound the number of large cliques and edge cliques in a yes-instance of the problem.

REDUCTION RULE 3.8. *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).*

Correctness proof. Let (A, B) be any monopolar partition of $V(G)$. Since a large clique C has size at least three, at least $|C| - 1 \geq 2$ vertices from C must belong to the same cluster in A. By Reduction Rule 3.7, the number of edges between any large clique and any other large or edge clique is at most one. It follows from the aforementioned statements that two vertices from two different large cliques, or from a large clique and an edge clique, must belong to different clusters in A. Consequently, if the number of large cliques in $(C_1, \ldots, C_r)$ is more than k, then for any monopolar partition (A, B) of G, the number of clusters in A is more than k, and hence (G, k) is a no-instance of MONOPOLAR RECOGNITION.

Suppose now that the number of large cliques in $(C_1, \ldots, C_r)$ is $\ell \leq k$ and that the number of edge cliques is ℓ'. From above, for any monopolar partition (A, B), no vertex from an edge clique can belong to a cluster in A containing a vertex from a large clique. Let C_i and C_j, $i < j$, be any two edge cliques. Since B is an independent set, at least one vertex from each edge clique must be in A. By property (iii) of a nice decomposition, no cluster in A can contain three vertices from three different edge cliques in $(C_1, \ldots, C_r)$. It follows from the aforementioned two statements that the number of clusters in A that contain vertices from edge cliques in $(C_1, \ldots, C_r)$ is at least $\ell'/2$. Now the set of clusters in A containing vertices from large cliques is disjoint from that containing vertices from edge cliques, and hence the number of clusters in A is at least $\ell+\ell'/2$. If the number of large cliques plus the number of edge cliques is more than $2k$, then $\ell+\ell' > 2k$, and hence $\ell+\ell'/2 \geq \ell/2+\ell'/2 > k$. This means that for any monopolar partition (A,B) of G, the number of clusters in A is more than k. It follows that (G,k) is a no-instance of MONOPOLAR RECOGNITION. ∎

Next, we sanitize the connections between already determined clusters in A_{true} and the remaining cliques in the normalized partition.

REDUCTION RULE 3.9. *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* $B_{\mathsf{true}} = B_{\mathsf{true}} \cup \{v\}$.

Correctness proof. To prove that the correctness of the reduction rule in case (1) holds, suppose that v is the only vertex in C_i that is adjacent to C. Let (A,B) be any monopolar partition of G. Let w be any vertex in C that is adjacent to v. Since C_i is a large clique, there exists a vertex $u \in C_i$, with $u \neq v$, such that $u \in A$. Since v is the only vertex in C_i that is adjacent to C, u is not adjacent to w. Now if v were in A, then since $C \subseteq A$ and hence $w \in A$, (u,v,w) would be an induced P_3 in A, contradicting that A is a cluster graph. It follows that $v \in B$ for any monopolar partition (A,B) of G.

To prove that the correctness of the reduction rule in case (2) holds, suppose that v is the only vertex in C_i that is not adjacent to C. Let (A,B) be any monopolar partition of G. Since C_i is a large clique, there exists a vertex $u \in C_i$, with $u \neq v$, such that $u \in A$. Since v is the only vertex in C_i that is not adjacent to C, u is adjacent to some vertex $w \in C$. Now if v were in A, then since $C \subseteq A$ and hence $w \in A$, (v,u,w) would be an induced P_3 in A, contradicting that A is a cluster graph. It follows that $v \in B$ for any monopolar partition (A,B) of G. ∎

Suppose that none of the above reduction rules applies to the instance (G,k). Then, the following lemma holds.

LEMMA 3.10. *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 such that* C_i *is adjacent to* C*. If* G *admits a monopolar partition, then* $C \cup C_i$ *induces a clique in* G*.*

Proof. Suppose, to get a contradiction, that $C \cup C_i$ does not induce a clique, and hence, there exists a vertex $x_i \in C_i$ such that x_i is not adjacent to some vertex in C. Since C and C_i are adjacent, there exist vertices $y_i \in C_i$ and $v \in C$ such that v and y_i are adjacent. Since Reduction Rule 3.5 is not applicable, x_i is not adjacent to any vertex in C, and y_i is adjacent to every vertex in C. Since cases (1) and (2) of Reduction Rule 3.9 are not applicable, there exist vertices $y'_i \neq y_i$ and $x'_i \neq x_i$ in C_i such that y'_i is adjacent to C and x'_i is not adjacent to C. Since Reduction Rule 3.5 is not applicable, y'_i is adjacent to every vertex in C. Now for any monopolar partition (A,B) of G, since B is an independent set, at least one vertex $w \in \{y_i, y'_i\}$ is in A, and at least one vertex of $u \in \{x_i, x'_i\}$ is in A. But then (v,w,u) is an induced P_3 in A, contradicting that A is a cluster graph. ∎

The above structure allows us to simplify the instance by shrinking already determined clusters in A_{true}.

REDUCTION RULE 3.11. *Let* (G,k) *be an instance of* MONOPOLAR RECOGNITION*, and let the tuple* $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ *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 $V_1 = \{u_C \mid C$ is a cluster in $A_{\mathsf{true}}\}$, $V_2 = \{v_1, \dots, v_{k+1}\}$, and $V_3 = C_1 \cup \dots \cup C_r$; 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 $C_1, \dots, C_r$ plus the $k+1$ new vertices, and keeping $C_1, \dots, C_r$ the same.

Correctness proof. To prove the correctness of the reduction rule, we need to show that (G, k) is a yes-instance of Monopolar Recognition if and only if (G', k) is. First, observe that by Reduction Rule 3.4, no vertex in $C_1 \cup \dots \cup C_r$ is adjacent to any vertex in B_{true}.

If $A_{\mathsf{true}} = \emptyset$, then the reduction rule consists of removing the vertices in B_{true} from G and replacing them with $k+1$ isolated vertices $v_1, \dots, v_{k+1}$. Since $A_{\mathsf{true}} = \emptyset$ and no vertex in $C_1 \cup \dots \cup C_r$ is adjacent to any vertex in B_{true}, the vertices in B_{true} are isolated vertices in G. Therefore, the reduction rule in this case essentially consists of removing some isolated vertices from B_{true} and G, and thus is obviously correct.

Assume now that $A_{\mathsf{true}} \neq \emptyset$. It is easy to see that if (G, k) is a yes-instance of Monopolar Recognition, then so is (G', k). This can be seen as follows. If (A, B) is a valid monopolar partition of (G, k), then the above reduction rules guarantee that $A_{\mathsf{true}} \subseteq A$, and hence each cluster of A_{true} must be a subset of a single cluster in A. If we (i) remove the vertices in B_{true} and add $k+1$ vertices to B that induce an independent set, and (ii) replace each cluster C in A_{true} by a single vertex u_C connected to the $k+1$ new vertices in B and to the vertices of the cluster that C belongs to A, we still get a valid monopolar partition of G.

To prove the converse, suppose that (G', k) is a yes-instance of Monopolar Recognition, and let (A', B') be a valid monopolar partition of $V(G')$. Since (A', B') is a valid monopolar partition of $V(G')$, and every vertex u_C (C is a cluster in A_{true}) is adjacent to the $k+1$ independent vertices $v_1, \dots, v_{k+1}$, we have $u_C \in A'$ for every cluster C in A_{true}, and $\{v_1, \dots, v_{k+1}\} \subseteq B'$. Let $B = B' \setminus \{v_1, \dots, v_{k+1}\} \cup B_{\mathsf{true}}$. Since (1) B' induces an independent set, (2) every vertex u_C (C is a cluster in A_{true}) is in A', and (3) no vertex in $C_1 \cup \dots \cup C_r$ is adjacent to any vertex in B_{true}, it follows that B is an independent set. Let A be the set of vertices obtained from A' by replacing each vertex u_C by the vertices in the cluster C in A_{true}. We claim that A is a cluster graph with at most k clusters. Suppose that a vertex u_C is replaced in A' by the vertices in cluster C in A_{true}; assume that u_C belongs to cluster C' in A'. Each vertex in C', other than u_C, must be a vertex in $V_3 = C_1 \cup \dots \cup C_r$. Let $v' \in C' \setminus \{u_C\}$ be chosen arbitrarily. Since v' and u_C belong to the same cluster C', by definition of G', v' must be adjacent to C in G. By Reduction Rule 3.5, v' must be adjacent to all vertices in C. Since v' was an arbitrarily chosen vertex in $C' \setminus \{u_C\}$, $(C' \setminus \{u_C\}) \cup C$ induces a cluster in A. It remains to show that no two clusters in A are adjacent. Suppose, to get a contradiction, that this is not the case. Since A' induces a cluster graph, there must exist two vertices u_{C_1} and u_{C_2} in A' that belong to clusters C'_1 and C'_2 in A', respectively, such that cluster $C_1 \cup (C'_1 \setminus \{u_{C_1}\})$ is adjacent to cluster $C_2 \cup (C'_2 \setminus \{u_{C_2}\})$. Since A' is a cluster graph, this implies that either (1) C_1 is adjacent to C_2, (2) C_1 is adjacent to $C'_2 \setminus \{u_{C_2}\}$, or (3) C_2 is adjacent to $C'_1 \setminus \{u_{C_1}\}$. This leads to a contradiction in each of the three cases above: (1) would contradict that A_{true} is a cluster graph (Reduction Rule 3.3), (2) would imply (by the construction of G') that u_{C_1}, and hence C'_1, is adjacent to C'_2 in A', and (3) would imply (by the construction of G') that u_{C_2}, and hence C'_2, is adjacent to C'_1 in A'. It follows from the above that the constructed partition (A, B) is a valid monopolar partition for $V(G)$. Finally, the number of clusters in A is the same as that in A', which is at most k. ∎

If Reduction Rule 3.11 is applied, then, after its application, we set A_{true} to V_1 and B_{true} to $\{v_1, \ldots, v_{k+1}\}$. Observe that this implies that $|A_{\mathsf{true}}| \leq k$ and $|B_{\mathsf{true}}| \leq k+1$. Note that in any valid monopolar partition (A, B) of the graph resulting from the application of Reduction Rule 3.11, 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 $V_1 \subseteq A$.

3.3. Modeling the pushing process by a bipartite graph. We have now arrived at a stage where we 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 to obtain a polynomial-size problem kernel. 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 describe the structure of the avalanche of pushes to A or B, we introduce the following auxiliary graph.

DEFINITION 3.12. *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 $V(\Lambda) = (V_C, V_I)$, 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 3.1. For a vertex $v \in V(\Lambda)$, we write $N_\Lambda(v) := N(v) \cap V(\Lambda)$ for the neighbors of v in Λ. To bound the maximum degree in Λ, we apply the following reduction rule. The correctness proof is straightforward, after we recall that the vertex cliques induce an independent set in G (Fact 3.1) and observe that no two vertices of an independent set can belong to the same cluster in a cluster graph.

REDUCTION RULE 3.13. *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 $A_{\mathsf{true}} = A_{\mathsf{true}} \cup \{v\}$.*

After performing all rules up to this point, we have the following lemma.

LEMMA 3.14. *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$, and consider the auxiliary graph $\Lambda = (V(\Lambda) = (V_C, V_I), E(\Lambda))$. 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 3.13 is inapplicable. By property (ii) of a nice decomposition and the inapplicability of Reduction Rule 3.6, every vertex clique that is adjacent to a large clique C is adjacent to exactly one vertex in C. Since by Reduction Rule 3.8 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 $|N_\Lambda(v)| \leq k$. ∎

Using the following lemma, we now observe that the auxiliary graph Λ captures some of the avalanches emanating from vertices in large or vertex cliques. Namely, pushing a vertex v in a large clique to B (or in a vertex clique to A) will also require pushing each vertex reachable (in the auxiliary graph) from v from A to B or vice versa.

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 $\bar{0}_n$ for the set of even integers in $\{0,\dots,n\}$, and write $\bar{1}_n$ for the set of odd integers in $\{0,\dots,n\}$.

LEMMA 3.15. *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1,\dots,C_r)$ be a normalized partition of $V(G)$, let $\Lambda = (V(\Lambda), E(\Lambda))$ be the associated auxiliary graph where $V(\Lambda) = (V_C, V_I)$, and let (A, B) be any valid monopolar partition of G.*

(i) *For any vertex $v \in V_C$: If $v \in B$, then $N_\Lambda(v) \subseteq A$.*
(ii) *For any vertex $v \in V_I$: If $v \in A$, then $N_\Lambda(v) \subseteq B$.*
(iii) *For any vertex $v \in V_C$: 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 any vertex $v \in V_I$: 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 $(C_1,\dots,C_r)$ and the inapplicability of Reduction Rule 3.6, v is not a neighbor of w in 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. □

The above lemma about the avalanches captured by the auxiliary graph allows us to identify vertices whose push to one side of the partition would lead to avalanches that, in turn, would lead to too many clusters in A or to two adjacent vertices in B. We can hence fix them in the corresponding part.

REDUCTION RULE 3.16. *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1,\dots,C_r)$ be a normalized partition of $V(G)$, and let $\Lambda = (V(\Lambda) = (V_C, V_I), E(\Lambda))$ 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 $B_{\mathsf{true}} = B_{\mathsf{true}} \cup \{v\}$.*

Correctness 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 3.15, 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) Let $v \in V_I$, and suppose that 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 vertices. If $v \in A$ for any valid partition (A, B) of G, then by part (iv) of Lemma 3.15, it would follow that $\bigcup_{i\in\bar{0}_n} N^i_\Lambda(v) \subseteq A$ and $\bigcup_{i\in\bar{1}_n} N^i_\Lambda(v) \subseteq B$. In either case this contradicts that (A, B) is valid partition of G: If $|\bigcup_{i\in\bar{0}_n} N^i_\Lambda(v)| > k$, then A contains more than k clusters since the vertices in $\bigcup_{i\in\bar{0}_n} N^i_\Lambda(v)$ induce an independent set in G, and if $\bigcup_{i\in\bar{1}_n} N^i_\Lambda(v)$ contains two adjacent vertices, then B is not an independent set. □

We are now ready to define a set of representative vertices which already capture the remaining structure of avalanches in the instance.

DEFINITION 3.17. *Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$, and let $\Lambda = (V(\Lambda) = (V_C, V_I), E(\Lambda))$ be the associated auxiliary graph. From each large clique C_i, $i \in [r]$, fix three vertices u_i, v_i, w_i, and 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 $V_{\mathsf{edge}} = \{C_i \mid C_i \text{ is an edge clique}\}$ 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 G that are also in $V(\Lambda)$, and define $N^{\cup}_{\mathsf{edge}} = \bigcup_{v \in N_{\mathsf{edge}}} \bigcup_{i \le n} N^i_\Lambda(v)$ to be the set of all vertices in $V(\Lambda)$ that are reachable in Λ from the vertices in N_{edge}. (Note that $N_{\mathsf{edge}} \subseteq N^{\cup}_{\mathsf{edge}}$.) 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 $N^{\cup}_{\mathsf{inter}} = \bigcup_{v \in V_{\mathsf{inter}}} \bigcup_{i \le n} N^i_\Lambda(v)$ to be the set of all vertices in $V(\Lambda)$ that are reachable in Λ from the vertices in V_{inter}. (Note that $V_{\mathsf{inter}} \subseteq N^{\cup}_{\mathsf{inter}}$.) Finally, let $V_{\mathsf{rep}} = A_{\mathsf{true}} \cup B_{\mathsf{true}} \cup V_{\mathsf{fixed}} \cup N^{\cup}_{\mathsf{inter}} \cup V_{\mathsf{edge}} \cup N^{\cup}_{\mathsf{edge}}$.*

The next reduction rule shrinks the instance to the set of representative vertices defined above.

REDUCTION RULE 3.18. *Let (G, k) be an instance of* MONOPOLAR RECOGNITION*, and let the tuple $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be a normalized partition of $V(G)$. Let V_{rep} be as defined in Definition* 3.17*. Set $G = G[V_{\mathsf{rep}}]$.*

Correctness proof. To prove the correctness of the reduction rule, let $G' = G[V_{\mathsf{rep}}]$, and we need to show that the two instances (G, k) and (G', k) of MONOPOLAR RECOGNITION are equivalent. Since G' is a subgraph of G and the property of having a valid monopolar partition is a hereditary graph property, it follows that if (G, k) is a yes-instance of MONOPOLAR RECOGNITION, then so is (G', k). Therefore, it suffices to show the converse, namely that if (G', k) is a yes-instance of MONOPOLAR RECOGNITION, then so is (G, k).

Suppose that (G', k) is a yes-instance of MONOPOLAR RECOGNITION, and let (A, B) be a valid monopolar partition of (G', k). Let $(A_{\mathsf{true}}, B_{\mathsf{true}}, C_1, \ldots, C_r)$ be the normalized partition of $V(G)$ with respect to which V_{rep}, and hence $G' = G[V_{\mathsf{rep}}]$, were defined, and let $V_{\mathsf{fixed}}, V_{\mathsf{inter}}, N^{\cup}_{\mathsf{inter}}, V_{\mathsf{edge}}, N^{\cup}_{\mathsf{edge}}$ be as in Definition 3.17. Let v be an arbitrary vertex in $V(G) \setminus V(G')$. It suffices to show that $G[V(G') \cup \{v\}]$ has a valid monopolar partition, as we can repeatedly add vertices, one after the other, and the same proof applies. Since the set of vertices forming the edge cliques, V_{edge}, is a subset of $V(G')$, and $A_{\mathsf{true}} \cup B_{\mathsf{true}} \subseteq V(G')$, either v is a vertex of a large clique of G that is not in V_{fixed}, or v is a vertex clique in G. We distinguish these two cases.

Case 1. $v \in V(C_i) \setminus V_{\mathsf{fixed}}$, for some large clique C_i, where $i \in [r]$. Since three vertices from C_i are in V_{fixed}, at least two of these vertices must belong to a cluster, say C'_i, in part A of the partition (A, B). Note that since $A_{\mathsf{true}} \subseteq A$, if C_i has a neighbor in A_{true}, which must be a neighbor of all the vertices in C_i, including v, by Lemma 3.10, then this neighbor must be in C'_i. We first claim that $C'_i \cup \{v\}$ is a clique. Observe that since C'_i contains two vertices from V_{fixed}, and hence from C_i, by Reduction Rule 3.7, C'_i does not contain any vertices from a large clique other than C_i or from an edge clique. Moreover, by property (ii) of the nice decomposition $(C_1, \ldots, C_r)$ and Reduction Rule 3.6, C'_i does not contain any vertex clique. Therefore, C'_i consists only of vertices in C_i, plus possibly a single vertex in A_{true} that must be adjacent to all the vertices in C_i. Since $v \in C_i$, it follows that $C'_i \cup \{v\}$ is a clique.

Let S be the set of vertex cliques in A, and note that S is an independent set. Define the following layered structure. The root of this structure is v. The first layer contains the set of vertices (possibly empty), denoted $N_1(v)$, that are the neighbors of v in S, that is, $N_1(v) = N(v) \cap S$; and the second layer contains the set of vertices, denoted $N_2(v)$, that are the neighbors in B of the vertices of $N_1(v)$, that is $N_2(v) = N(N_1(v)) \cap B$. For $i > 2$, layer i contains the set of vertices $N_i(v) = N(N_{i-1}(v)) \cap S$ if i is odd, and the set of vertices $N_i(v) = N(N_{i-1}(v)) \cap B$ if i is even. Let $N_0(v) = \{v\}$. We claim that the partition (A', B') obtained from (A, B) by placing v in A, moving the vertices in $N_i(v)$ for even $i \geq 2$ from B to A, and moving the vertices in $N_i(v)$ for odd i from A to B, is a valid monopolar partition; that is, (A', B'), where $A' = (A \cup \bigcup_{i \in \bar{0}_n} N_i(v)) \setminus \bigcup_{i \in \bar{1}_n} N_i(v)$ and $B' = (B \cup \bigcup_{i \in \bar{1}_n} N_i(v)) \setminus \bigcup_{i \in \bar{0}_n} N_i(v)$, is a valid monopolar partition of $G[V(G') \cup \{v\}]$. Since S is an independent set, so is $\bigcup_{i \in \bar{1}_n} N_i(v) \subseteq S$. Since the set of neighbors of $\bigcup_{i \in \bar{1}_n} N_i(v)$ is precisely $\bigcup_{i \in \bar{0}_n} N_i(v)$ and B is an independent set, $B' = (B \cup \bigcup_{i \in \bar{1}_n} N_i(v)) \setminus \bigcup_{i \in \bar{0}_n} N_i(v)$ is an independent set as well. Therefore, to show that (A', B') is a valid monopolar partition, it suffices to show that A' is a cluster graph of at most k clusters.

First, we claim that each vertex in $N_{\mathsf{even}} = \bigcup_{i \in \bar{0}_n} N_i(v)$ belongs to a large clique in $C_1, \ldots, C_r$. This is certainly true for the vertex v, which is in C_i, where C_i is a large clique. Now for any other vertex $u \in N_{\mathsf{even}}$, by construction, u is the neighbor of a vertex clique in S. Since the set of all vertex cliques induces an independent set, u itself cannot be a vertex clique, being a neighbor of a vertex clique. Since $u \in N_i(v)$, for some i, and hence v is reachable from u, u cannot be an endpoint of an edge clique; otherwise, v would belong to $N^{\cup}_{\mathsf{edge}}$ and, hence, would belong to V_{rep}. Since $A_{\mathsf{true}} \subseteq A$, and no vertex in $V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$ is adjacent to a vertex in B_{true} by Reduction Rule 3.4, $u \notin A_{\mathsf{true}} \cup B_{\mathsf{true}}$. It follows from the preceding that each vertex in N_{even} belongs to a large clique in $C_1, \ldots, C_r$. From each large clique in $C_1, \ldots, C_r$, at least two fixed vertices are in A'; denote by C'_j the cluster in A' that contains the two fixed vertices from a large clique C_j. As shown at the beginning of Case 1 about C'_i, the same holds true for any C'_j: C'_j consists of a subset of C_j, plus possibly a vertex in A_{true} that is adjacent to all vertices in C_j. Now add each vertex in N_{even} that belongs to a (large clique) C_j in G to the corresponding cluster C'_j in A'. We claim that the resulting partition is a valid monopolar partition. Since each vertex u in N_{even} was added to the cluster C'_j such that $u \in C_j$ and C'_j consists of a subset of C_j plus possibly a vertex in A_{true} that is adjacent to all vertices in C_j, $C'_j \cup \{u\}$ is a clique. Moreover, since each vertex $u \in N_{\mathsf{even}}$ was added to an existing cluster, this addition does not increase the number of clusters in A, and hence, A' has at most k clusters. It remains to show that this addition does not create an edge between two different clusters. Suppose that this is not the case. Since $N_{\mathsf{even}} \subseteq B$ is an independent set, this implies that there exists a vertex $u \in N_{\mathsf{even}}$ that is added to a cluster C'_j in A such that u is adjacent to some vertex w in A. Since all the neighbors of u that are vertex cliques are in $\bigcup_{i \in \bar{1}_n} N_i(v) \subseteq B'$, w is not a vertex clique. Since v is reachable from u, and hence from w, and $v \notin N^{\cup}_{\mathsf{edge}}$, w cannot be a vertex of an edge clique. By the same token, since v is reachable from w and $v \notin N^{\cup}_{\mathsf{inter}}$, w cannot be a vertex of a large clique. Finally, w cannot be in B_{true} because no vertex in $V(G) \setminus (A_{\mathsf{true}} \cup B_{\mathsf{true}})$ is adjacent to a vertex in B_{true}, and w cannot be in A_{true} because w would be adjacent to all vertices of C'_j. This completes the proof of Case 1.

Case 2. v is a vertex clique. The treatment of this case is very similar to Case 1. We define $N_0(v) = \{v\}$, $N_i(v) = N(N_{i-1}(v)) \cap B$ if $i \geq 1$ is odd, and $N_i(v) = N(N_{i-1}(v)) \cap S$ if $i \geq 2$ is even, where S is the set of vertex cliques in A. It can then be shown—using very similar arguments to those made in Case 1—that the partition (A', B'), where $A' = (A \cup \bigcup_{i \in \bar{1}_n} N_i(v)) \setminus \bigcup_{i \in \bar{0}_n} N_i(v)$ and $B' = (B \cup \bigcup_{i \in \bar{0}_n} N_i(v)) \setminus \bigcup_{i \in \bar{1}_n} N_i(v)$, is a valid monopolar partition of $G[V(G') \cup \{v\}]$. The proof is omitted to avoid repetition. ∎

With this reduction rule we have finally bounded the size of $V(G)\setminus A_{\mathsf{true}} \cup B_{\mathsf{true}}$ and may now give the polynomial kernel whose existence was promised in Theorem 1.1.

THEOREM 3.19. MONOPOLAR RECOGNITION *has a polynomial kernel with at most* $9k^4 + 9k + 1$ *vertices which can be computed in* $\mathcal{O}(n^2m)$ *time.*

Proof. Given an instance (G, k) of MONOPOLAR RECOGNITION, we apply Reduction Rules 3.3–3.18 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 3.3–3.18 applies, and let $\Lambda = (V(\Lambda) = (V_C, V_I), E(\Lambda))$ be the auxiliary graph. Note that, by Reduction Rule 3.18, $V(G') = V_{\mathsf{rep}} = A_{\mathsf{true}} \cup B_{\mathsf{true}} \cup V_{\mathsf{fixed}} \cup N^{\cup}_{\mathsf{inter}} \cup V_{\mathsf{edge}} \cup N^{\cup}_{\mathsf{edge}}$. By Reduction Rule 3.8, 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 3.13, v has at most k neighbors in V_I. Moreover, by Reduction Rule 3.7, 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 3.16 does not apply and $\Delta(\Lambda) \leq k$ by Lemma 3.14, for any $v \in V(\Lambda)$ we have that $|\bigcup_{i \leq n} N^i_\Lambda(v)| \leq \Delta(\Lambda) \cdot k \leq k^2$. 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 3.7, 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 $k+k+1+3k+k^4+4k+8k^4 \leq 9k^4+9k+1$. Hence, the number of edges in G' is $\mathcal{O}(k^8)$, and thus the kernel has polynomial size.

It remains to show the running time. First, observe that the overall number of applications of the reduction rules is $\mathcal{O}(n)$, since each application either moves a vertex to A_{true} or B_{true}, or reduces the number of vertices in G. To obtain the overall running time bound, we first bound the time to check the applicability of each reduction rule.

For Reduction Rules 3.3–3.13, it is obvious that their applicability can be checked in $\mathcal{O}(m)$ time (recall that we assume $n \in \mathcal{O}(m)$).

For Reduction Rule 3.6, its applicability can be checked in $\mathcal{O}(m)$ time, if we assign to each vertex a label indicating the number of its clique and an additional "counter-variable" for each cluster. Then, we consider the vertices of the clique decomposition one by one. When considering a vertex v, we reset all clique counters to 0. Then we scan through the adjacency list of v, and increase the counter of a clique C_i for each edge between v and C_i (the cluster for each edge can be checked in $\mathcal{O}(1)$ time using the clique labels of the vertices). After scanning through the adjacency list, we check, for each clique C_i that v is adjacent to, whether the number of edges between v and C_i and the size of C_i meet the conditions in Reduction Rule 3.6.

For Reduction Rule 3.7, we create once in $\mathcal{O}(n^2)$ time an $n' \times n'$ matrix M where n' is the number of large and edge cliques. Entry $M[i,j] = M[j,i]$ is used to count the number of edges between the ith large or edge clique C_i and the jth large or edge clique C_j. Before and after we test the applicability of the rule, $M[i,j] = 0$ for all $i, j \in [n']$. To test applicability, we scan through a list containing each edge of G exactly once and increment $M[i,j]$ each time we encounter an edge between C_i and C_j. If at some point in this procedure $M[i,j] = 2$ for some i and j, then the rule applies. After the check, we reset M to 0 by keeping a list of all pairs of modified matrix indices.

It is clear that we can check in $\mathcal{O}(m)$ time whether Reduction Rule 3.8 applies.

Reduction Rule 3.9 can be checked in a similar manner to Reduction Rule 3.7. We use an $n_1 \times n_2$ matrix M', where n_1 is the number of large cliques in the current normalized partition and n_2 is the number of clusters in A_{true}. We use entry $M[i,j]$ to count the number of vertices in large clique C_i adjacent to the jth cluster D_j in A_{true}. After a one-time $\mathcal{O}(n^2)$ initialization, we will ensure that before and after the test of applicability $M[i,j] = 0$ for all $i \in [n_1]$, $j \in [n_2]$. Additionally, we use a vertex labeling for all vertices in G, which we initialize for every vertex as *uncounted*. We iterate in $\mathcal{O}(m)$ time over the list of edges in G and whenever we encounter an edge, one endpoint v of which is uncounted and in large clique C_i, and the other endpoint is in the jth cluster in A_{true}, then we increment $M[i,j]$ and remove the labeling from v. If, after processing some edge, $M[i,j]$ now equals 1, or $|C_i| - 1$, then we label C_i as *amenable* and otherwise remove the amenable-label from C_i (if any). Reduction Rule 3.9 applies if and only if a large clique is amenable after processing each edge. As before, after the check for applicability, we restore all entries $M[i,j] = 0$ by tracking the pairs of indices which changed during the applicability test.

Reduction Rule 3.11 can obviously be checked in $\mathcal{O}(m)$ time. For the remaining reduction rules, it is necessary to compute the auxiliary bipartite graph Λ, which can be done in $\mathcal{O}(m)$ time by iterating over all edges and checking whether they are incident with a large clique or vertex clique. To check whether Reduction Rule 3.16 applies, it is enough to compute the connected components of Λ, and compute for each component the size of each part and the subgraph of G that is induced by each part. This can clearly be done in $\mathcal{O}(m)$ time. For Reduction Rule 3.18, we first need to compute V_{rep} in $\mathcal{O}(m)$ time—we iterate over all edges and check whether one of the corresponding conditions applies to the endpoints—and then compute the subgraph $G[V_{\mathsf{rep}}]$ also in $\mathcal{O}(m)$ time.

The time to perform each reduction rule is $\mathcal{O}(m)$, plus the time needed to update the clique decomposition. We update the clique decomposition $\mathcal{O}(n)$ times; by Lemma 3.2, this takes $\mathcal{O}(nm)$ time. Thus, the latter step has a total running time of $\mathcal{O}(n^2 m)$, which gives the overall running time for computing the kernel. ∎

4. Kernel-size lower bound.

This section is dedicated to proving the "only if" direction of Theorem 1.1, which, together with Theorem 3.19, completes the proof of Theorem 1.1. More precisely, we prove the following.

THEOREM 4.1. *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* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, CLUSTER-$\Pi$-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 4.1. We show Theorem 4.1 by giving a cross-composition from the NP-hard problem COLORFUL INDEPENDENT SET [15], defined below.

COLORFUL INDEPENDENT SET
Input: A graph $G = (V, E)$, $k \in \mathbb{N}$, and a proper k-coloring $c \colon V \to \{1, \ldots, k\}$.
Question: Is there 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 the case of properties Π as above 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 pushing a vertex of M to the cluster graph side A. The crucial point here is that, because the forbidden induced subgraphs have order at least three, there are at least two possibilities for choosing a vertex from M to push. Now each distinct push of a vertex of M from B to A 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.

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. The gadgets are combined in a way that gives a trivial cluster-Π partition with one caveat: The overall number of clusters in $G[A]$ is one more than allowed, and there is exactly one singleton cluster which can be pushed to the Π-side B. We call the vertex of this singleton cluster an *activator* vertex since it activates the instance selection 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 the instance selection gadget will be pushed from A to B. This vertex corresponds to one Colorful Independent Set instance, and its push transmits the choice of this instance to further gadgets that check whether this instance has an independent set of k vertices.

Next, in section 4.1, we give an example of a problem instance that shows these binary-tree-like pushes—we later generalize the construction underlying this instance and use it in selection gadgets. Based on this intuition, we outline the construction in section 4.2. In section 4.3, we begin with the concrete description of the cross-composition: We give scaffolds for the construction and some basic operations that we need, such as fixing vertices to one of the two parts of the partition, and invariants that we need to maintain for the correctness proofs. Afterwards, the construction proceeds in an incremental fashion: In sections 4.4 and 4.5, we show how to construct a selection gadget and then use it to create instance-selection and vertex-selection gadgets, and we add them to the constructed graph one by one. Finally, in section 4.6 we construct verification gadgets that ensure that the selected vertices in the selected instance form an independent set.

4.1. Example. Figure 2 shows an example of the instance-selection gadget for Cluster-Π-Partition with Π being the set of cluster graphs. The forbidden induced subgraph for both A and B is the P_3 in this case, and in the example we look for a cluster-Π partition with at most four clusters in $G[A]$. Observe that, as highlighted in Figure 2, we build an instance with some initial cluster-Π partition (A, B) where $G[A]$ has one cluster more than allowed.

In the example, we aim to select one of four instances (G_1, k), (G_2, k), (G_3, k), and (G_4, k), and the selection of instance (G_i, k), for $i \in [4]$, is triggered—as shall be seen later—by a push of the vertex w_i from A to B. We ensure that exactly one of these vertices is pushed to B as follows. Of the initially five clusters in $G[A]$, all except one contain a special vertex called *anchor vertex* a_i^2. (The superscript is not necessary here and is used only for consistency with the formal construction given later on.) We add vertices to B (shown above each anchor vertex) to ensure that an anchor vertex a_i^2 cannot be moved to B: Pushing a_i^2 to B creates five P_3s that only intersect in a_i^2 and are otherwise independent. Consequently, to destroy these

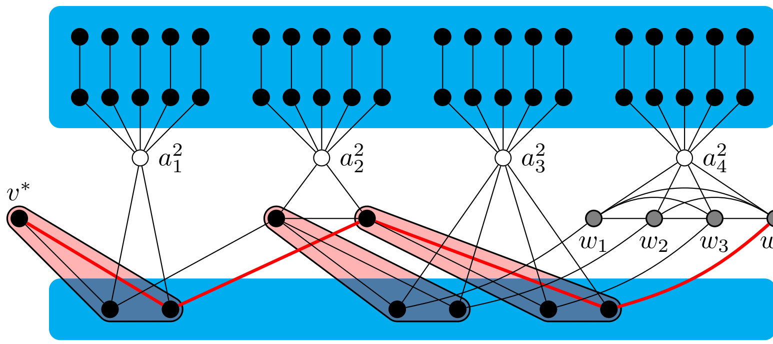

FIG. 2. *An example of the instance selection in the case where* Π *is the set of cluster graphs and one of four instances needs to be selected. The blue background highlights vertices that are initially in* B*. Anchor vertices are white. The vertices* w_1*,* w_2*,* w_3*, and* w_4*, corresponding to the instances from which one is to be selected, are grey. A possible path of pushes starting with* v^* *and ending with* w_4 *is drawn with thick red edges. The red regions with solid outline show the* P_3*s that we use as forbidden induced subgraphs for* $G[B]$ *in the argument that at least one* w_i*-vertex is pushed to* B*. (Color available online.)*

forbidden induced subgraphs in B, one needs to push five independent vertices to A, which would create too many clusters. Hence, the only viable push to decrease the number of clusters in $G[A]$ is pushing v^* from A to B. This, however, creates a P_3 in B since v^* has two nonadjacent neighbors in B. Consequently, one of these two neighbors needs to be pushed to A. Now this vertex is adjacent to the anchor vertex a_1^2 and to a vertex in the cluster containing the anchor vertex a_2^2. Since a_1^2 cannot be pushed to B as discussed above, we need to push the other vertex from A to B to ensure that $G[A]$ is P_3-free. This results in a new P_3 in $G[B]$, and again we have two possibilities to destroy this P_3 by pushing a vertex to A. In each case, the vertex that is pushed to A is adjacent to a_3^2 and to a vertex in the cluster of a_4^2. This vertex will be pushed to A. Depending on the possible choices of pushes from B to A, this vertex could be w_1, w_2, w_3, or w_4.

4.2. Construction outline. Let t instances of COLORFUL INDEPENDENT SET with graphs $G_1, \ldots, G_t$ be given; we will refer to the instances by their indices $1, \ldots, t$. The construction is divided into three main parts:

- an instance selection part, whose purpose is selecting one of the t COLORFUL INDEPENDENT SET instances;
- a vertex selection part, whose purpose is selecting from each color in the selected instance a vertex into the independent set; and
- a verification part, whose purpose is ensuring that the selected vertices form an independent set.

All these parts share a common scaffold, which is given by the so-called anchor vertices, as outlined below. See Figure 3 for a schematic view of the construction. For illustrative purposes, by (A, B) we refer to a cluster-Π partition for the constructed instance. Next, we explain the different components of the construction.

Anchors. The construction starts in section 4.3 by introducing a special set of vertices, which we call anchors, introduced before any of the selection and verification gadgets. We then introduce small gadgets that ensure that the cluster-Π partition (A, B) has the property that each anchor is part of a distinct cluster in the cluster-graph side $G[A]$ and that each cluster contains an anchor (similar to the ex-

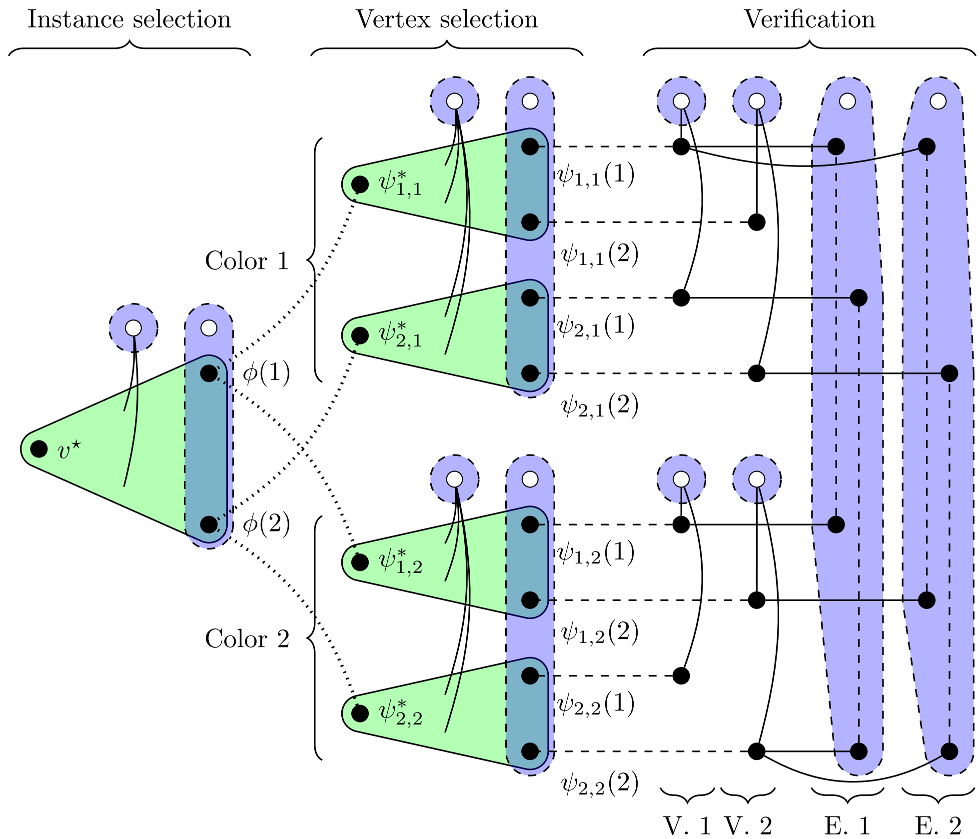

Fig. 3. *Schematic of the arrangement of the gadgets for modeling the instance and vertex selection and how dials are shared among them. In this example, there are two instances of* Colorful Independent Set*. Each instance has two color classes, two vertices of each color class, and two edges. Referring to the instance-selection part, the green solid area represents a selection gadget similar to the example given in section* 4.1 *(it is produced by a procedure called* selection*, given in section* 4.4*). The vertex on the left tip is the activator vertex, and the vertices on the right side are the choice vertices. White vertices represent anchors, and blue dashed areas represent dials (the corresponding edges are omitted). (Color available online.) We show only two anchors and dials in the selection gadget, but more may be present. Referring to the vertex-selection part, dotted edges indicate an additional construction that implies that if one of the endpoints is in B, then the other is as well; this transfers the choice of instance from the instance to the vertex selection gadgets. The vertex-selection part consists of one vertex-selection gadget for each color class and instance. Thus, in total there are four selection gadgets. The dials corresponding to each color are shared among the selection gadgets of that color, regardless of their instance. Referring to the verification part, dashed edges indicate an additional construction that implies that not both endpoints can be in B; we say that they have been made exclusive. This additional construction transfers the choice of vertices from the vertex-selection part to the verification part. The verification part consists of vertex gadgets (left, labeled V.* 1 *and V.* 2*) and edge gadgets (right, labeled E.* 1 *and E.* 2*). Their construction is explained in section* 4.6*. In essence, pushing a choice vertex to B in a vertex-selection gadget necessitates pushing a corresponding vertex to B in all edge-verification gadgets corresponding to adjacent edges. For example, putting $\psi_{1,1}(1) \in B$ requires that the corresponding vertices in edge gadgets to the right are in B. Edges are represented by pairs of exclusive vertices in the edge-verification gadgets. Thus, not both choice vertices corresponding to the endpoints of an edge can be put into B.*

ample given in section 4.1). In other words, each cluster in $G[A]$ "grows around" some anchor. The remaining gadgets will be attached to the anchors and contain different parts of the clusters around an anchor. We introduce the anchors at the beginning since they need to be *shared* by several gadgets. The reason is that the number of anchors will be equal to the number of clusters, the parameter in the constructed instance, which we need to keep small in order for the cross-composition to work.

Instance selection. In section 4.4, we introduce an instance-selection gadget, in analogy to the example from section 4.1. This gadget is represented by the leftmost triangle in Figure 3. Two of its main features are the activator vertex (on the left tip) and the set of choice vertices (on the right side). In the example in section 4.1, the choice vertices were labeled w_1 to w_4. The choice vertices correspond in a one-to-one fashion to the instances of Colorful Independent Set given as input. The activator vertex is not adjacent to any anchor, and since the number of clusters in $G[A]$ equals the number of anchors, the activator vertex v^* cannot be in A. By the properties explained in section 4.1, this means that one of the choice vertices is in B; this corresponds to selecting one instance. The remaining gadgets verify that the selected instance is a yes-instance.

Vertex selection. The next set of gadgets, described in section 4.5, aims to select a vertex from each color of the previously selected instance. This works similarly to the instance selection, by introducing a gadget—analogous to the example from section 4.1—for each color and each instance. These gadgets are shown in the center of Figure 3. As before, each gadget consists of an activator vertex (on its left tip), and a set of choice vertices (on its right side). For each instance $i \in [t]$ and color $\ell \in [k]$, the choice vertices in the gadget instance G_i correspond in a one-to-one fashion to the vertices of color ℓ in G_i.

The choice of instance—as done before—is transferred to the vertex-selection gadgets using a small additional gadget that has the following effect: If a choice vertex of some instance is in B, then in *each* vertex-selection gadget for that instance (i.e., for all colors) the activator vertex is in B. This construction is indicated by the dashed lines in Figure 3. Again, by the properties of the selection gadget explained in section 4.1, this means that, for each color ℓ, there is a vertex in G_i that is selected by a choice vertex in the gadget for color ℓ.

A crucial part to the cross-composition construction is obtaining an upper bound on the number of clusters in $G[A]$ that is polynomial in k, the independent set size, and logarithmic in t, the number of instances. Note that each of the selection gadgets uses a number of clusters, and hence, it is imperative (for the reasons explained in the previous sentence) that the selection gadgets share clusters. Figure 5 shows schematically which gadgets share clusters. The corresponding cliques will be merged into one large clique containing exactly one anchor. We will call these large cliques *dials*, supporting the intuition that exactly one of the gadgets sharing a dial may be active in a cluster-Π partition.

Verification. In the final part of the construction, given in section 4.6, we ensure that it is impossible to simultaneously push into B two choice vertices that correspond to two adjacent vertices in some input instance. By itself, this would be easy to do: Simply introduce a forbidden induced subgraph for Π that contains the two choice vertices. However, we also need to allow pushing *one* of the two choice vertices into B. Since a vertex corresponding to a choice vertex may be adjacent to many other vertices in the corresponding input graph, the corresponding parts of the forbidden subgraphs may overlap and inadvertently introduce new forbidden subgraphs. Hence, we need a slightly more involved construction that consists of more independent vertex and edge gadgets, shown schematically on the right in Figure 3; the construction is described in detail in section 4.6.

The basic idea is to have an edge gadget for each edge $e = \{uv\}$ in an COLORFUL INDEPENDENT SET instance and vertex gadgets corresponding to the endpoints u, v of that edge. If the choice vertex corresponding to endpoint u (resp., v) is in B, then vertex gadgets and additional gadgetry will ensure that a vertex, u_e (resp., v_e), corresponding to u (resp., v) in the edge gadget is in B as well. Using additional gadgetry we make u_e and v_e part of a forbidden subgraph for Π whose remaining vertices are fixed in B—this ensures that not both u_e and v_e can be in B, that is, not both u and v are included in the independent set.

4.3. Scaffolding and basic subconstructions. We now begin with the formal description of the cross-composition and prove its properties. Let t instances of COLORFUL INDEPENDENT SET be given, with graphs $G_1, \dots, G_t$, respectively. Below, we use an instance and its index in $[t]$ interchangeably. Without loss of generality, assume that the following properties, which can be achieved by simple padding techniques, hold:

- Each instance asks for an independent set of size k (otherwise, introduce new colors and isolated vertices as needed);
- each color class in each graph has n vertices, and n is a power of two (otherwise, in a color class that does not satisfy this property, add as needed new vertices that are adjacent to all vertices in all other color classes); and
- t is a power of two (otherwise, duplicate one of the instances as needed).

In the following, let m be the maximum number of edges over all graphs G_i for $i \in [t]$.

We construct an instance of CLUSTER-Π-PARTITION as described in this section and the following sections. 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). Throughout, we denote by (A, B) an arbitrary fixed cluster-Π partition of the (so-far constructed) graph G. We also fix M to be a forbidden induced subgraph of Π with a minimum number of vertices. By the properties of Π, M contains at least three vertices. 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 show below how to achieve that. We then successively add gadgets that are attached to these anchors, as outlined in section 4.2.

4.3.1. Anchors. As mentioned before, the construction begins by adding *anchor* vertices. In section 4.3.2, we will add gadgets to ensure that each anchor vertex is in A. We introduce d anchor vertices, divided into $4 + 2k$ groups, as follows:

- Introduce the anchors a_1^1, a_2^1; these two anchors serve to fix vertices into B by making any such vertex adjacent to both a_1^1 and a_2^1.
- Introduce the anchors $a_1^2, a_2^2, \dots, a_{2\log t}^2$; these anchors will be used in the instance-selection gadget.
- Introduce the anchors $a_1^3, a_2^3, \dots, a_{k+1}^3$; these anchors serve to connect the instance-selection gadget with the vertex-selection gadgets.
- For each $i \in [k]$, introduce the anchors $a_1^{3+i}, a_2^{3+i}, \dots, a_{\log n}^{3+i}$; these anchors are used by the vertex-selection gadgets.
- For each $i \in [k]$, introduce the anchors $a_1^{3+k+i}, a_2^{3+k+i}, \dots, a_n^{3+k+i}$; these anchors are used in vertex subgadgets of the verification gadgets.

- Finally, introduce the anchors $a_1^{4+2k}, a_2^{4+2k}, \ldots, a_m^{4+2k}$; these anchors are used in edge subgadgets of the verification gadgets.

Hence, we define the number d of desired clusters to be $d \coloneqq 2 + 2\log(t) + (k+1) + k\log n + kn + 2m$.

4.3.2. Helper, dial, and volatile vertices, fixing anchors. Before explaining how to ensure that all anchors are in A, we introduce some notation. Many of the gadgets will fix some vertices into A or B using some additional auxiliary vertices. To avoid having to reason about the fixed and auxiliary vertices, we will name them and introduce invariants of the ensuing construction that allow us to ignore these vertices when constructing cluster-Π partitions.

Throughout, we use the following notation. 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. On the other hand, volatile vertices are normally in B and may be pushed to A. To gain some intuition, consider Figure 2. The vertices in the top blue area will be helper vertices, the vertices on white background in the middle (except for v^*) will be dial vertices, the vertices on blue background on the bottom will be volatile vertices.

We now 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. See Figure 2 for the special case of $d = 5$ and M being a P_3. Suppose that $a_i^j \in B$. 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 is 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. More generally, we maintain the following invariant throughout the construction.

INVARIANT 4.2. *For each cluster-Π partition (A, B) of G with at most d clusters in $G[A]$, all anchors are in A and all helper vertices are in B. Each helper vertex was introduced as part of a copy of M and made adjacent to both a_1^1 and a_2^1. No helper vertex has any neighbors other than those in the copy of M it was introduced in and the anchors a_1^1 and a_2^1.*

4.3.3. Dials and joining dials. As mentioned before, the gadgets that we are about to construct share clusters in $G[A]$. Hence, for a shared cluster, we need to ensure that its parts coming from different gadgets are mutually and completely adjacent. To describe the gadgets in a self-contained way, we define dials: A *dial* will be a set D of vertices that contains exactly one anchor, say a, and that induces a clique. At some point in the construction, we may want to construct a new gadget that uses the anchor a. That is, the gadget needs to contain a subset S of vertices that shall form a cluster with a, together with all other vertices that have already/previously been added to the cluster containing a. To do so, we let *S join the dial D*, whose anchor is a, by making $D \cup S$ a clique.

Formally, we associate each anchor a_i^j with a vertex set D_i^j containing a_i^j. (And D_i^j is supposed to induce a clique in G throughout the construction.) We say that D_i^j is the *dial* of a_i^j. Initially, we have $D_i^j = \{a_i^j\}$. Later on, other vertices may join D_i^j, which is defined as follows. By making a vertex *v join* a dial 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. Vertex v is then designated as a dial vertex. Throughout, we maintain the following invariant.

INVARIANT 4.3. (i) *Each dial induces a clique in G.* (ii) *No two dial vertices in different dials are adjacent.*

Note that this is true so far, since the only vertices currently in dials are anchors.

Next, there will be two different flavors of dials, similar to what we showed in section 4.1: A dial may serve either to push a vertex from this dial into B or to accept a (volatile) vertex which is pushed to A. In order to simplify the reasoning about which vertices may be in a cluster in $G[A]$, we use the following invariant.

INVARIANT 4.4.

(i) *The following dials D_i^j are singletons:* (a) *for each $j \in \{2\} \cup \{4, 5, \dots, 3+k\}$ and each odd i (these are the dials used in the instance- and vertex-selection gadgets), and* (b) *for each $j \in \{3+k+1, \dots, 3+2k\}$ (and each i; these are the dials used in the vertex subgadgets of the verification gadgets).*

(ii) *For each anchor a_i^j, each volatile vertex is either nonadjacent to a_i^j or adjacent to all vertices in a_i^j's dial D_i^j.*

A particular corollary will be that each cluster in $G[A]$ either contains an anchor a_i^j with a dial $D_i^j = \{a_i^j\}$, or contains only vertices of D_i^j. This will help in the correctness proof, where we build a cluster-Π partition for G piece-by-piece.

We introduce the following terminology:

DEFINITION 4.5 (friendly partition). *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 $G[A]$.*

4.3.4. Making vertices exclusive. As a final prerequisite, we need the operation of making three vertices u, v, w exclusive. Intuitively, this operation was the main tool used in section 4.1 to fan out the possible pushes in avalanches according to a binary-tree-like structure: 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 gadget described below.

DEFINITION 4.6 (making three vertices exclusive). *Let $u, v, w \in V(G)$ be three vertices satisfying the following conditions: At most two of u, v, w are dial vertices. Furthermore, any edge between two vertices $\{u, v, w\}$ is contained in a dial.*

By making u, v, and w exclusive *we mean*

(i) *introducing a copy of (the forbidden subgraph) M consisting of new vertices into G;*

(ii) *identifying three distinct vertices of M with u, v, and w—if there are two dial vertices among u, v, and w, then we identify them with two adjacent vertices of M; and*

(iii) *making each remaining vertex (if any) of M (different from u, v, and w) adjacent to both anchors a_1^1 and a_2^1.*

The vertices in $V(M) \setminus \{u, v, w\}$ are helper vertices.

See Figure 4 for an illustration.

Observe that step (iii) entails that $V(M) \setminus \{u, v, w\} \subseteq B$, since otherwise there would be a P_3 in $G[A]$ involving a_1^1 and a_2^1. Hence, Invariant 4.2 is maintained by this operation (clearly, Invariant 4.4 is maintained as well). Invariant 4.3 is maintained as well since all dial vertices among u, v, and w are contained in the same dial by the preconditions on these vertices.

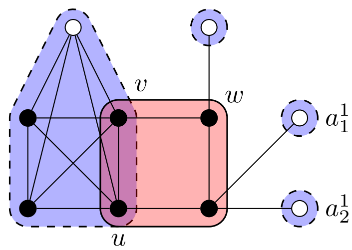

FIG. 4. *Making u, v, and w exclusive. In this example, M is a cycle with four vertices, u and v are part of a dial, and w is adjacent to some anchor. Dials are represented by blue regions with dashed outlines and the introduced copy of M by the red region with solid outline. Anchor vertices are white. (Color available online.)*

Furthermore, not all three $u, v, w \in B$; otherwise, since $V(M) \setminus \{u, v, w\} \subseteq B$, $G[B]$ would contain a copy of M. That is, making vertex exclusive indeed imposes the constraint on (A, B) that we are aiming for. For further reference, we state this fact in the following lemma.

LEMMA 4.7. *Let G be the graph obtained at any point during the construction in which $u, v, w \in V(G)$ were made exclusive, and let (A, B) be a cluster-Π partition for G with at most d clusters in $G[A]$. Then at least one of the vertices u, v, w is not in B.*

To simplify arguing about the existence of cluster-Π partitions, we will always tacitly assume that $V(M) \setminus \{u, v, w\} \subseteq B$ and ignore the vertices in $V(M) \setminus \{u, v, w\}$. To simplify proving that the constructed partitions are indeed cluster-Π partitions, we derive the following sufficient conditions.

LEMMA 4.8. *Let G be the graph obtained at any point during the construction in which $u, v, w \in V(G)$ were made exclusive using a copy of M, denoted as M in a slight abuse of notation, and let (A, B) be a bipartition of $V(G)$. Suppose that*
(i) *$G[A]$ is a cluster graph with at most d clusters,*
(ii) *$G[B \setminus V(M)] \in \Pi$,*
(iii) *at least one of u, v, w is in A, and*
(iv) *u, v, w are each adjacent only to some subset of $\{u, v, w\}$ in $G[B]$.*
Then, $(A, B \cup (V(M) \setminus \{u, v, w\}))$ is a cluster-Π partition for G.

Proof. Without loss of generality, by symmetry, we may assume that $u \in A$. Clearly, it suffices to prove that $G[B \cup (V(M) \setminus \{u, v, w\})] \in \Pi$. Suppose that there is a copy M' of M contained in $G[B \cup (V(M) \setminus \{u, v, w\})]$ as an induced subgraph. Since $G[B \setminus V(M)] \in \Pi$, graph M' contains a vertex of $V(M)$. Since $u \in A$ and thus $|V(M) \setminus A| < |V(M')|$, graph M' moreover contains a vertex of $B \setminus V(M)$. By condition (iv), v and w are adjacent in $G[B]$ only to some subset of $\{v, w\}$. Since M is connected, there is an edge e in M' between $V(M) \setminus \{v, w\}$ and $B \setminus V(M)$. Indeed, since $u \in A$, edge e contains a vertex of $V(M) \setminus \{u, v, w\}$, that is, a helper vertex. By Invariant 4.2, helper vertices in M do not receive further edges, and e thus contains a helper vertex and an anchor vertex. Note that each anchor vertex a is in A: Each of a's incident copies of M consists otherwise only of helper vertices, and by Invariant 4.2 these copies are present in G as well. Thus, if a were in B, then there would be more than d clusters in $G[A]$, a contradiction to assumption (i). Thus, e contains a vertex in A, a contradiction to the fact that e is in M', which is a subgraph of $G[B]$. ∎

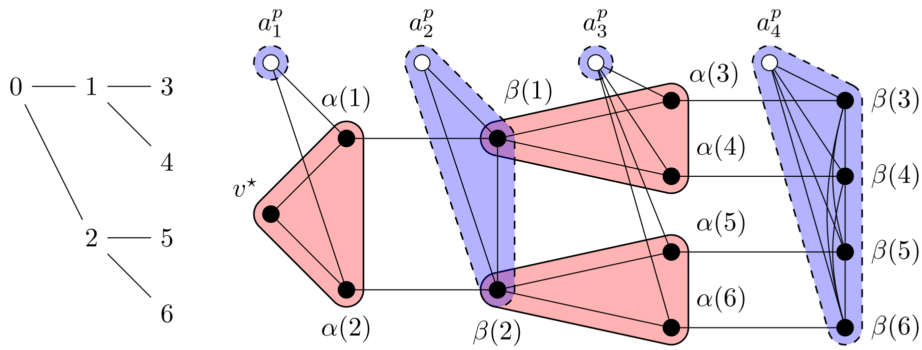

FIG. 5. *Left: An example for the tree T used in* selection(p, q) *for $q = 4$. Right: Parts of the selection gadget constructed by* selection(p, q) *using T where $q = 4$ and M is a P_3. Vertex $v^\star$ is the activator vertex, and vertices $\beta(3)$ through $\beta(6)$ are the choice vertices. Copies of M are highlighted with a red region with solid outline; that is, the corresponding vertices have been made exclusive. The parts of the dials of the anchors that are used in the construction are highlighted with a blue region with dashed outline. Not shown are the gadgets used for fixing the anchors in A and parts of the dials that possibly were previously constructed. (Color available online.)*

The operation of making vertices exclusive can naturally be applied to (only) two vertices.

DEFINITION 4.9. *Let $u, v \in V(G)$ be two vertices such that, if they both are dial vertices, then they are contained in the same dial. By* making u and v exclusive *we mean*

(i) *introducing a new helper vertex x;*
(ii) *making x adjacent to both a_1^1 and a_2^1; and*
(iii) *making u, v, and x exclusive.*

Note that Lemmas 4.7 and 4.8 hold analogously.

We next explain a generic selection gadget construction and then use it to construct an instance-selection gadget and, for each instance and color, vertex-selection gadgets.

4.4. Generic selection gadget and instance selection.

We now introduce a generic selection gadget that we use for both instance and vertex selection. The inner workings of the gadget use the necessary pushes along a binary-tree-like structure as outlined in section 4.1; refer to Figures 2 and 5 for examples. That is, the gadget is constructed such that it allows a trivial cluster-Π partition (A, B) for the resulting graph with $d+1$ clusters in $G[A]$, one more than allowed. This cluster is a singleton, called the activator vertex. Pushing it to B results in a forbidden induced subgraph in $G[B]$ requiring subsequent pushes to A. Each of these pushes to A will create a P_3 involving two anchors, meaning that the third vertex has to be pushed to B. This again creates a forbidden subgraph in B, and so on. The leaves in the resulting tree-like structure correspond to the selection to be made. That is, there is a set of dial vertices, which we call *choice* vertices below, which are normally in A. Through a path of pushes in the binary-tree-like structure, one of the choice vertices will be pushed to B. This push will in turn activate other gadgets.

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; see Figure 5. 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.

Construction. We use the following (generic) construction, called $\mathsf{selection}(p, q)$, 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 is a power of two. For example, in the instance-selection gadget we will set $p = 2$ and $q = t$. Refer to Figure 5 for an example of the construction.

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, proceed as follows. Introduce two vertices $\alpha(v), \beta(v)$ into G. Let i be the level of v. Connect $\alpha(v)$ to both a^p_{2i-1} and $\beta(v)$. Make $\beta(v)$ join D^p_{2i}. Next, 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 dial vertex. Each $\alpha(v)$, $v \in V(T)$, is a volatile vertex. Each $\beta(v)$, $v \in V(T)$, is a dial vertex.

We now verify that the above construction maintains all invariants. Observe that Invariant 4.2 is maintained as none of the previously introduced helper vertices receive new edges. Invariant 4.3(i) is maintained: Each dial still induces a clique, because each introduced dial vertex joined some dial, except for v^* which is not adjacent to any other dial vertex. Invariant 4.3(ii) is maintained as well, because each dial vertex is made adjacent either only to some vertices of one specific dial, or to nondial vertices. Invariant 4.4(i) is maintained since no vertex joins the referenced dials. The only volatile vertices that have been introduced are the vertices $\alpha(v)$, $v \in V(T)$. These vertices have been made adjacent to only one anchor a^p_ℓ where ℓ is odd. By Invariant 4.4(i), a^p_ℓ's dial is a singleton, and thus Invariant 4.4(ii) is maintained.

Denote the constructed gadget as $\mathsf{selection}(p, q)$, 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 4.10. *Let G' be the graph before applying* $\mathsf{selection}(p, q)$*, and let G be 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 $\mathcal{D}$ for some dial set $\mathcal{D}$, then (A, B) is friendly with respect to the dials $\mathcal{D}$.*

(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 $G[B]$.*

Proof. (i) Note that there are d anchors, and each anchor is in A. Hence, each cluster in $G[A]$ consists of an anchor and possibly further vertices. By assumption, we have $v^* \in B$. We now prove by induction that for each $i \in [\log q]$, there is at least one vertex $v \in L_i$ with $\beta(v) \in B$, yielding the statement. Consider the case $i = 1$. Let $u, v \in L_1$. As $v^* \in B$, we have that either $\alpha(u)$ or $\alpha(v)$ is in A; say $\alpha(u) \in A$, and the other case is symmetric. Since $\alpha(u)$ is adjacent to both a^2_{2i-1} and $\beta(u)$, we have $\beta(u) \in B$ as, otherwise, vertices $a^2_{2i-1}, \alpha(u), \beta(u)$ would form an induced P_3 in $G[A]$. That is, the statement holds if $i = 1$. Now suppose that for some $u \in L_{i-1}$, $i > 1$, we have $\beta(u) \in B$. Consider the children v, w of u in T. Since $\beta(u), \alpha(v), \alpha(w)$ are made exclusive, either $\alpha(v)$ or $\alpha(w)$ is in A. Say $\alpha(v) \in A$ and the other case is symmetric. Note that $\alpha(v)$ is adjacent to both a^2_{2i-1} and $\beta(v)$. Hence, $\beta(v) \in B$ since, otherwise, $a^2_{2i-1}, \alpha(v), \beta(v)$ would induce a P_3 in $G[A]$. Thus, indeed, for some $v \in L_i$ we have $\beta(v) \in B$.

(ii) For the second statement, let (A', B') be a cluster-Π partition for G'. Construct a cluster-Π partition (A, B) for G as follows. Put $(A, B) = (A', B')$. Put $v^* \in A$. For each $v \in T$ at level $i > 0$, put $\alpha(v) \in B$ and $\beta(v) \in A$. This concludes the construction. Clearly, each choice vertex is in A, as required.

We claim that $G[A]$ is a cluster graph with $d + 1$ clusters. Note that v^* is not adjacent to any vertex in A and hence constitutes a singleton cluster. By Invariant 4.4, each anchor a_i^j whose dial D_i^j is not a singleton is contained in a cluster in $G[A]$ whose vertex set is contained in D_i^j. Apart from v^*, the only vertices from the construction placed into A are contained in dials which are not singletons, and hence, $G[A]$ is a cluster graph with $d+1$ clusters. From this fact it is also immediate that, if (A, B) is a cluster-Π partition and (A', B') is friendly with respect to the dials $\mathcal{D}$, then (A, B) is friendly with respect to the dials $\mathcal{D}$.

To conclude the proof of statement (ii), we apply Lemma 4.8 to show that $G[B] \in \Pi$. Note that all vertices in $B \setminus B'$ are part of a triple of vertices that has been made exclusive by $\mathsf{selection}$. Furthermore, no two vertices between two different triples have been made adjacent by $\mathsf{selection}$. Thus, it is enough to show that the conditions in Lemma 4.8 are satisfied. Note that each triple of exclusive vertices contains one vertex from A. Furthermore, for each vertex $v \in T$, $\alpha(v)$ is connected in B only to $\alpha(w)$ where w is the sibling of v in T. Thus, $G[B] \in \Pi$.

(iii) For the third statement, let (A', B') be a cluster-Π partition for G'. Given $i \in [q]$, we construct a cluster-Π partition (A, B) for G as follows (as before, we ignore helper vertices). Set $A = A'$, $B = B'$, and note that $v^* \in B$. Pick a path P in T from the root r to the leaf v_ℓ corresponding to the ith choice vertex; call it $\beta(v_\ell)$. For each vertex $v \in V(T) \setminus \{r\}$, if $v \in V(P)$, put $\alpha(v) \in A$ and $\beta(v) \in B$. Otherwise, if $v \notin V(P)$, put $\alpha(v) \in B$ and $\beta(v) \in A$. Clearly, $\beta(v_\ell) \in B$ and $\beta(v_\ell)$ is isolated in $G[B]$, as required.

We first show that $G[A]$ is a cluster graph with at most d clusters. Suppose that

$G[A]$ contains an induced P_3, say Q. Clearly, Q contains at least one vertex introduced by the construction selection. As all helper vertices are in B, path Q does not involve helper vertices. By Invariant 4.3, Q involves a volatile vertex; that is, $\alpha(v) \in V(Q)$ for some $v \in V(T)$. Moreover, $v \in V(P)$ as otherwise $\alpha(v) \in B$. By construction, apart from helper vertices $\alpha(v)$ is adjacent in G only to $\beta(v)$, $\alpha(w)$ (where w is v's sibling in T), and a^p_{2i-1}, where i is v's level in T. As $\beta(v), \alpha(w) \in B$ by definition of (A, B), path Q contains a^p_{2i-1}. As D^p_{2i-1} is a singleton by Invariant 4.4, that is, $D^p_{2i-1} = \{a^p_{2i-1}\}$, and since (A', B') is friendly with respect to the dials D^p_{2i-1}, we have that D^p_{2i-1} is a singleton cluster in $G[A']$. Recall that, by construction, the only new vertices adjacent to a^p_{2i-1} are vertices $\alpha(x)$ for $x \in L_i$. By definition of (A, B), only one of these vertices $\alpha(x)$ is in A, namely $\alpha(v)$. Hence, $\alpha(v)$ is the only neighbor of a^p_{2i-1} in $G[A]$, a contradiction to Q being an induced P_3 in $G[A]$. To see that there are at most d clusters, observe that each vertex in A is adjacent to one of the anchors, and thus, there are at most d connected components.

To show that $G[B] \in \Pi$ by Lemma 4.8 it remains to show that each triple of three vertices that were made exclusive include one vertex in A, and that they are adjacent in $G[B]$ only to some subset of themselves (apart from helper vertices). By construction, the only triples of exclusive vertices are $\beta(u)$, $\alpha(v)$, $\alpha(w)$ for some $u \in V(T)$ and its children v, w. (The case of v^* is analogous.) Either v or w is not in $V(P)$, and hence, either $\alpha(v)$ or $\alpha(w)$ is in A, that is, at least one vertex in the triple is in A, as required. It remains to show the condition on their adjacencies. If $\beta(u) \in B$, then $\alpha(u) \in A$, and, hence, regardless of whether $\beta(u) \in B$, a possible connection outside of the triple must involve $\alpha(v)$ or $\alpha(w)$; say it involves $\alpha(v)$. Vertex $\alpha(v)$ is only adjacent to $\beta(u)$, to some anchor, and to $\beta(v)$. If $\alpha(v) \in B$, then $\beta(v) \in A$. Thus, indeed, $\beta(u)$, $\alpha(v)$, and $\alpha(w)$ are only connected within themselves in $G[B]$ (apart from helper vertices). This shows that (A, B) is a cluster-Π partition with d clusters in $G[A]$. ∎

Instance selection. As mentioned before, to construct the *instance-selection gadget*, we apply selection$(2, t)$. 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.

4.5. Vertex selection. We apply selection to create vertex-selection gadgets for each instance and each color. Each vertex-selection gadget selects one vertex of the gadget's color into the independent set when activated by putting its 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. See Figure 3 for illustration.

In the first part of the construction of the vertex-selection gadgets, for each instance $r \in [t]$ and color $i \in [k]$, we carry out selection$(3 + i, n)$ to introduce the *vertex-selection gadget* for instance r and color i. 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 $\psi_{r,i}$ corresponds in a one-to-one fashion to the vertices of color i in instance r. This is how the selection of an independent-set vertex is modeled.

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. See Figure 6 for an illustration. To achieve this, for each $r \in [t]$, we carry out the following steps. Introduce a volatile vertex v_r. Make $\phi(r)$ and v_r exclusive.

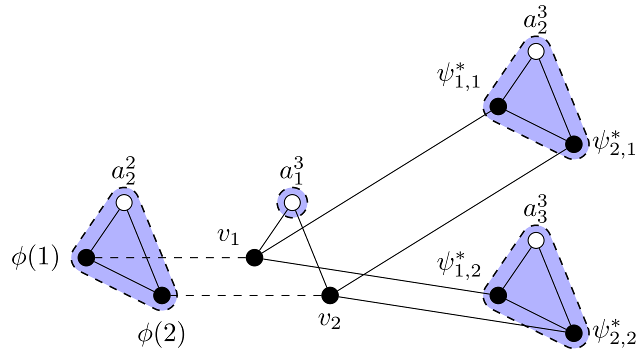

FIG. 6. *Illustration of the second part of the construction of the vertex-selection gadgets in which we connect them to the instance-selection gadget. In this example, there are two instances, represented by the choice vertices $\phi(1)$ and $\phi(2)$ of the instance-selection gadget. Each instance has two colors; $\psi_{1,1}^*$, $\psi_{2,1}^*$, $\psi_{1,2}^*$, and $\psi_{2,2}^*$ are the activator vertices of the corresponding vertex-selection gadgets. Anchor vertices are white. Dials are shown by blue regions with dashed outlines. Dashed edges mean that the endpoints have been made exclusive. (Color available online.)*

Make v_r adjacent to a_1^3, and, for each $i \in [k]$, make v_r adjacent to $\psi_{r,i}^*$. This concludes the construction of the vertex-selection gadgets.

Intuitively, the selection of instance r is indicated by placing $\phi(r) \in B$. Since $\phi(r)$ and v_r are exclusive, $v_r \in A$. Vertex v_r forms a P_3 with a_1^3 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 from each color.

We now verify after this construction that the invariants are maintained. Invariant 4.2 is maintained because it is maintained by the operations of selection and making vertices exclusive. Invariants 4.3 and 4.4 are maintained in the first part of the construction because selection maintains these invariants. In the second part of the construction, no dial vertices are added, giving Invariant 4.3 and Invariant 4.4(i). Invariant 4.4(ii) holds for a_1^3 since D_1^3 is a singleton. For all the other anchors Invariant 4.4(ii) holds because the invariant was satisfied before the second part of the construction, and because each volatile vertex v_r is only made adjacent to the single anchor a_1^3. Thus, the construction of the vertex-selection gadgets maintains Invariants 4.2 to 4.4.

LEMMA 4.11. *Let G be the graph after constructing the vertex-selection gadgets.*

(i) *If G admits a cluster-Π partition (A, B) with d clusters in $G[A]$, then there is an instance $r \in [t]$ such that, for each color $i \in [k]$, there is at least one vertex $v \in V(G_r)$ of color i satisfying that $\psi_{r,i}(v) \in B$.*

(ii) *For each instance $s \in [t]$ and each vertex subset $V' \subseteq V(G_s)$ containing exactly one vertex of each color, there is a cluster-Π partition (A, B) for G such that $G[A]$ contains at most d clusters, $\psi_{s,i}(V') \subseteq B$, and all other choice vertices of each vertex-selection gadget are in A. Moreover, the choice vertices that are contained in B are isolated in $G[B]$.*

Proof. (i) There are d anchors in $G[A]$, and the activator vertex of the instance-selection gadget is not adjacent to any of the anchors. Thus, the activator vertex is in B. By Lemma 4.10(i), it follows that, for at least one instance $r \in [t]$, we have $\phi(r) \in B$. Since $\phi(r)$ and v_r are exclusive, we have $v_r \in A$. Since, for each $i \in [k]$, vertices a_1^3, v_r, and $\psi_{r,i}^*$ form a P_3, we have that $\psi_{r,i}^* \in B$ for each $i \in [k]$. By

Lemma 4.10(i), it follows that, for each $i \in [k]$, there is one choice vertex of the ith vertex-selection gadget that is in B. Thus, for each $i \in [k]$, there is a vertex $v \in V(G_r)$ of color i such that $\psi_{r,i}(v) \in B$, as required.

(ii) Without loss of generality, assume that the instance-selection gadget has been constructed first, and the vertex-selection gadgets have been constructed in ascending order of instances and then colors. We start by showing that a partial cluster-Π partition with the required properties exists after the first part of the construction, and then we proceed to treat the second part. For the first part, we show that a suitable partial cluster-Π partition exists after each call to selection.

Let G_0 be the graph obtained after introducing the instance-selection gadget. Before introducing any selection gadget, the graph has a trivial cluster-Π partition with d clusters in A that is friendly with respect to each dial. By Lemma 4.10(iii), there is a cluster-Π partition (A_0, B_0) for G_0 such that $G_0[A_0]$ has d clusters, and out of all choice vertices only the rth one is in B. Furthermore, this cluster-Π partition is friendly with respect to the dials D_ℓ^{3+i}, $i \in [k]$.

In the following, let $s \in [t]$ be the instance for which we want to construct a cluster-Π partition. Let G_{s-1} be the graph obtained after introducing all vertex-selection gadgets for instances in $[s-1]$. By iteratively applying Lemma 4.10(ii), starting with G_0 and (A_0, B_0), we obtain that there is a cluster-Π partition (A_{s-1}, B_{s-1}) for G_{s-1} such that, for each vertex-selection gadget, each activator vertex is in A (in a cluster together with the dial it joined) and each choice vertex is in A. Since we joined the activator vertices to some dials, $G_{s-1}[A_{s-1}]$ has d clusters. Moreover, since (A_0, B_0) is friendly with respect to the dials D_ℓ^{3+i}, $i \in [k]$, (A_{s-1}, B_{s-1}) is friendly with respect to these dials as well.

Let $V' \subseteq V(G_s)$ as in the statement of the lemma. For each $i \in [k]$, denote by $v'_i \in V'$ the vertex of color i in V', and let $G_{s,i}$ be the graph obtained after introducing the vertex-selection gadget for instance s and color i (in the first part of the construction of the vertex-selection gadgets). By induction on i and by Lemma 4.10(iii), we obtain that $G_{s,i}$ admits a cluster-Π partition $(A_{s,i}, B_{s,i})$ with d clusters in $G_{s,i}[A_{s,i}]$ such that, for each $j \in [i]$, we have $\psi^*_{s,j} \in B$, $\psi_{s,j}(v'_j) \in B$, and such that all other choice vertices in any vertex-selection gadget are in A. Moreover, $(A_{s,i}, B_{s,i})$ is friendly with respect to the dials D_ℓ^{3+j}, $j \in \{i, i+1, \ldots, k\}$ (whence we can apply induction).

Let G_t be the graph obtained after introducing all vertex-selection gadgets for instances in $[t] \setminus [s-1]$. By applying iteratively Lemma 4.10(ii) to $G_{s,k}$ and $(A_{s,k}, B_{s,k})$, we obtain a cluster-Π partition (A_t, B_t) for G_t analogously to the cluster-Π partition for G_{r-1}. Hence, the statement of the lemma holds after the first part of the construction of the vertex-selection gadgets. It remains to incorporate the second part, that is, to incorporate vertices v_r, $r \in [t]$, into (A_t, B_t). Construct a cluster-Π partition (A, B) for G from (A_t, B_t) as follows. Put $A = A_t$, $B = B_t$. For each $r \in [t] \setminus \{s\}$, put $v_r \in B$. Finally, put $u_s \in B$ and $v_s \in A$.

We claim that $G[A]$ is a cluster graph with at most d clusters. Recall that $G_t[A_t]$ contains at most d clusters (corresponding to the d anchors). Thus, $G[A]$ has at most d connected components since, for each $r \in [t]$, vertex v_r is connected to some anchor. To show that $G[A]$ does not contain an induced P_3, it is enough to show that, for each $r \in [t]$, either $v_r \in B$ or, for all $i \in [k]$, $\psi^*_{r,i} \in B$. The fact that $v_r \in B$ is trivial for $r \neq s$; otherwise, if $r = s$, we have $\psi^*_{r,i} \in B$ by the construction of $(A_{s,i}, B_{s,i})$.

Note that, for each $r \in [t]$, either $\phi(r)$ or v_r is in A. Hence, by Lemma 4.8 (and Definition 4.9), to show that $G[B] \in \Pi$, it suffices to prove, for each $r \in [t]$, the property that vertices $\phi(r)$ and v_r are adjacent in $G[B]$ only to each other (apart

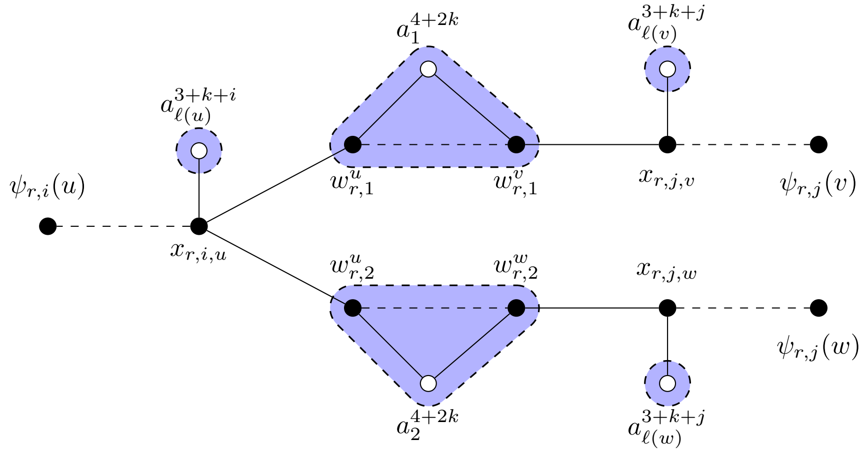

FIG. 7. *Parts of the verification gadget for three vertices and two edges in instance r. There are three vertices u, v, and w in instance r. Vertex u is of color i, and vertices v and w are of color j (their indices that are used in the construction are denoted by $\ell(u), \ell(v), \ell(w)$, respectively). There are two edges, $e_1 = \{u, v\}$ and $e_2 = \{u, w\}$. Dials are highlighted with blue regions with dashed outlines. A dashed edge means that its endpoints have been made exclusive—only one of the endpoints can be in B. (Color available online.)*

from helper vertices). If $r \neq s$, we have $v_r \in A$ and, thus, by construction of (A_0, B_0) according to Lemma 4.10(iii), that $\phi(r)$ is an isolated vertex in $G_0[B_0]$, giving the required property. If $r = s$, then $\phi(s) = \phi(r) \in A$. For the incident edges of v_r, by construction of $(A_{s,i}, B_{s,i})$, $i \in [k]$, according to Lemma 4.10(ii), for each $i \in [k]$ we have $\psi^*_{s,i} \in A$; that is, none of the nonhelper neighbors of v_r is in B. Thus, indeed, by Lemma 4.8, $G[A] \in \Pi$, finishing the proof. ∎

4.6. Verification. We now construct the verification gadgets. It is again crucial that gadgets share clusters (anchors) in order to keep the overall number of clusters in A small. How the clusters are shared is indicated in Figure 3. In essence, we fix a vertex ordering for each color and each instance, and an edge ordering for each instance. Then, for each color, we use one vertex gadget that represents all the first vertices of that color, one vertex gadget that represents all the second vertices of that color, and so on. Similarly, for the edges, the first edge gadget represents all the first edges of each instance, the second edge gadget represents all the second edges, and so on.

The working principle of the gadgets is as follows. See Figures 3 and 7 for illustration. Each vertex gadget consists of a singleton dial and a vertex for each instance that could be pushed into that dial. Selecting a vertex v via a vertex-selection gadget will make it necessary to push the vertex corresponding to v into the cluster containing the dial of its vertex gadget. Next, each edge gadget consists of a dial and, for each instance, two vertices corresponding to the endpoints of an edge in that instance. The push of a vertex to the dial of a vertex gadget creates a P_3 in A for each incident edge e, necessitating further pushes. Namely, we are required to push a vertex out of the dial of the edge gadget in A representing 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. This is achieved by making the two corresponding vertices in the constructed graph exclusive.

The formal construction is as follows. See Figure 7 for an illustration. For each $r \in [t]$, let $E(G_r) = \{e_1, \ldots, e_m\}$. (If there are fewer than m edges, duplicate an arbitrary edge as needed.) For each $j \in [m]$, perform the following steps towards constructing the jth *edge gadget*. Let $e_j = \{u, v\}$. Introduce two vertices $w^u_{r,j}, w^v_{r,j}$ into G. Make $w^u_{r,j}$ and $w^v_{r,j}$ exclusive. Make $w^u_{r,j}$ and $w^v_{r,j}$ join D^{4+2k}_j (they are thus dial vertices).

We furthermore need for each vertex a *vertex gadget*, which is constructed for each instance $r \in [t]$ and each color $i \in [k]$ as follows. Fix an arbitrary ordering of the vertices of color i in G_r and say the *index* of a vertex is its index in that ordering. For each vertex $v \in V(G_r)$ of color i, introduce a vertex $x_{r,i,v}$ into G. Make $\phi_{r,i}(v)$ and $x_{r,i,v}$ exclusive. Make $x_{r,i,v}$ adjacent to a^{3+k+i}_ℓ, where ℓ is the index of v. Vertex $x_{r,i,v}$ is a volatile vertex.

Finally, connect the edge gadgets and vertex gadgets as follows. For each instance $r \in [t]$, perform the following steps. Recall that $E(G_r) = \{e_1, \ldots, e_m\}$. For each $j \in [m]$, let $i_1, i_2 \in [k]$ be the colors of the endpoints $v_1, v_2 \in V(G_r)$ of e_j. Make x_{r,i_1,v_1} adjacent to $w^{v_1}_{r,j}$, and make x_{r,i_2,v_2} adjacent to $w^{v_2}_{r,j}$. This finishes the construction of the verification gadgets and concludes the construction of the graph G in our instance of Cluster-Π-Partition.

Let us now verify that the above construction maintains the invariants. Clearly, Invariant 4.2 remains valid since making vertices exclusive maintains Invariant 4.2 and otherwise no new helper vertices are introduced. Invariant 4.3 remains valid since only the vertices $w^u_{r,j}$ and $w^v_{r,j}$ join a dial and each pair joins the same dial. Invariant 4.4(i) remains valid since it was valid before and no vertex joins any dial of the form D^{3+k+i}_ℓ. Invariant 4.4(ii) remains valid since the only vertices made adjacent to anchors are $x_{r,i,v}$ and the corresponding dial is a singleton by Invariant 4.4(i).

Lemma 4.12. *Let G be the graph constructed above. The graph G admits a cluster-Π partition (A, B) with d clusters in $G[A]$ if and only if there exists an instance $s \in [t]$ such that G_s has an independent set with exactly one vertex of each color.*

Proof. Assume that G admits a cluster-Π partition (A, B) with d clusters in $G[A]$. Note that Lemma 4.11 refers to a subgraph of G, the graph resulting from constructing all the vertex-selection gadgets. By restricting (A, B) to that subgraph, from Lemma 4.11(i) we infer that there is an instance $s \in [t]$ such that, for each color $i \in [k]$, there is a vertex $v_i \in V(G_s)$ such that $\psi_{s,i}(v_i) \in B$. We claim that $V' := \{v_i \mid i \in [k]\}$ is an independent set in G_s. Suppose V' is not an independent set, and let $e_j \in E(G_s)$ be such that $e_j \subseteq V'$. Let $e_j = \{u, v\}$, and let i, i' be the colors of u and v, respectively. Since $\psi_{s,i}(u)$ and $x_{s,i,u}$ are exclusive, we have $x_{s,i,u} \in A$. Thus, $w^u_{s,j} \in B$ as, otherwise, a^{3+k+i}_ℓ, $x_{s,i,u}$, and $w^u_{s,j}$ would form an induced P_3 in $G[A]$, where ℓ is the index of u. Similarly, $w^v_{s,j} \in B$. However, $w^v_{s,j}$ and $w^u_{s,j}$ are exclusive, and each of them is contained in B. This contradicts the fact that $G[B] \in \Pi$. Hence, V' is an independent set. Clearly, V' contains exactly one vertex of each color.

Now assume that, for some instance $s \in [t]$, there is an independent set $V' = \{v_i \mid i \in [k]\} \subseteq V(G_s)$ with exactly one vertex v_i of each color $i \in [k]$. Let G' be the graph obtained in the construction before constructing the verification gadgets. By Lemma 4.11(ii), there is a cluster-Π partition (A', B') for G' with d clusters in $G'[A']$ such that, for each $i \in [k]$, we have $\psi_{s,i}(v_i) \in B'$ (and these vertices are isolated in $G[B']$), and all other choice vertices of each vertex-selection gadget are in A'.

We now construct a cluster-Π partition (A, B) for G from (A', B'). Put $A = A'$ and $B = B'$. For each instance $r \in [t]$ including s, and for each $v \in V(G_r)$, let i be the color of v. If v is not in the independent set V', then put $x_{r,i,v} \in B$, and if $v \in V'$, then put $x_{r,i,v} \in A$ instead. For each edge $e_j \in E(G_r)$, and each of its endpoints, $v \in e_j$, if $v \notin V'$, then put $w^v_{r,j} \in A$, and if $v \in V'$, then put $w^u_{r,j} \in B$. Clearly, not both endpoints can be in the independent set.

Observe that (A, B) is a bipartition of $V(G)$. We claim that (A, B) is a cluster-Π partition for G with at most d clusters in $G[A]$. We first show that $G[A]$ is a cluster graph. Suppose that $G[A]$ contains an induced P_3, say Q. Since $G'[A']$ is a cluster graph, Q contains a vertex in $V(G) \setminus V(G')$. By Invariant 4.3, Q involves a nondial vertex, that is, a vertex v from a vertex gadget. Since $v \in A$, by definition of (A, B), we have $v = x_{s,i,v_i} \in V(Q)$ for some $v_i \in V'$. The only neighbors of x_{s,i,v_i} in G are $\psi_{r,i}(i)$ and $w^{v_i}_{r,j}$, where $j \in J$ for some set $J \subseteq [m]$ (apart from helper vertices). By definition of (A, B), each of these vertices is in B, a contradiction to the existence of Q. Hence, $G[A]$ is a cluster graph. To see that $G[A]$ contains at most d connected components, observe that $G'[A']$ has at most d connected components, one for each anchor, and each vertex in $A \setminus A'$ is connected to an anchor in $G[A]$.

It remains to show that $G[B] \in \Pi$. Recall that $G'[B'] \in \Pi$. The only edges in G between vertices in $V(G')$ and newly introduced vertices in $V(G) \setminus V(G')$ are incident with either an anchor or some choice vertex of some vertex-selection gadget. The anchors are in A, and if some of the choice vertices are in B', then they are isolated in $G'[B']$ by Lemma 4.11(ii). Thus, it is enough to show that these choice vertices and the newly introduced vertices induce a subgraph of G that satisfies Π. Since all of these vertices have been made exclusive, it is enough to show that the conditions of Lemma 4.8 are satisfied for each pair that has been made exclusive. Each such pair has the form (i) $(w^u_{r,j}, w^v_{r,j})$ or (ii) $(\psi_{r,i}(v), x_{r,i,v})$. By definition of (A, B), out of each pair, at least one vertex is in A. Thus, it remains to prove the adjacency condition of Lemma 4.8. As $\psi_{r,i}(v)$, if contained in B, is a singleton in $G'[B]$, by construction, there is no edge in $G[B]$ between any two pairs of form (ii). There is no edge between two pairs of form (i) since, by definition of B, for each edge gadget $j \in [m]$, there is exactly one pair of form (i) containing a vertex in B, and there is no edge between any two pairs of form (i) for distinct edge gadgets j. Finally, there is no edge in $G[B]$ between two pairs of forms (i) and (ii): Assume there is such an edge e, and let $v \in V(G_r)$ and $j \in [m]$ correspond to the two pairs. By construction, e is between two pairs of forms (i) and (ii) that correspond to the same instance r (otherwise, no edge has been introduced between them). Moreover, $v \in e_j$ for $e_j \in E(G_r)$. That is, $e = \{w^v_{r,j}, x_{r,i,v}\}$. We have $w^v_{r,j} \in B$ only if $v \in V'$. However, $x_{r,i,v} \in A$ by definition, and, thus, $e \nsubseteq B$. Thus, the conditions of Lemma 4.8 are satisfied, meaning that $G[B] \in \Pi$. It follows that (A, B) is the required cluster-Π partition. □

It is not hard to verify that the construction can be carried out in polynomial time. Since $d \leq \text{poly}(\log t + \max_{i=1}^{t} |V(G_i)|)$, we thus have shown that all the conditions of cross-compositions are satisfied, yielding Theorem 4.1.

5. Kernels for parameterization by the size of one of the parts. In this section, we prove that (Π_A, Π_B)-Recognition has a polynomial kernel parameterized by the size of one of the parts of the bipartition when Π_A and Π_B satisfy certain general technical conditions. To simplify the presentation, we pick B to be the part whose size is at most the parameter k. We then consider the conditions that Π_A is characterized by forbidden induced subgraphs, each of size at most d, and Π_B is hereditary (closed under taking induced subgraphs). In the first subsection, we give a polynomial kernel with $\mathcal{O}(d!\,(k+1)^d)$ vertices in this general setting. In the second subsection, we consider the restricted setting of Cluster-Π_Δ-Partition: Π_A is the set of all cluster graphs (P_3-free graphs), and Π_B a hereditary property that contains only graphs of degree at most Δ. Although the result of the first subsection implies a kernel with $\mathcal{O}(k^3)$ vertices in this setting, we prove that CLUSTER-Π_Δ-PARTITION actually has a kernel with $\mathcal{O}((\Delta^2+1)k^2)$ vertices.

5.1. A kernel in the generic setting. In this subsection, we prove that (Π_A, Π_B)-RECOGNITION has a polynomial kernel with $\mathcal{O}((d+1)!\,(k+1)^d)$ vertices when Π_A can be characterized by forbidden induced subgraphs, each of size at most d, and Π_B is hereditary. We obtain the kernel by applying a powerful lemma of Fomin, Saurabh, and Villanger [20] that is based on the Sunflower Lemma (see [10, 17]).

Let $\mathcal{U}$ be a universe, and let $\mathcal{F}$ be a set of subsets of $\mathcal{U}$. Recall that $X \subseteq \mathcal{U}$ is a *hitting set* of $\mathcal{U}$ if $X \cap F \neq \emptyset$ for every $F \in \mathcal{F}$.

LEMMA 5.1 (see [20, Lemma 2]). *Let $\mathcal{U}$ be a universe, and let k be an integer. Let $\mathcal{F}$ be a set of subsets of $\mathcal{U}$, each of size at most d. Then in $\mathcal{O}(|\mathcal{F}|\,(k+|\mathcal{F}|))$ time, we can find a set $\mathcal{F}' \subseteq \mathcal{F}$ of size at most $d!\,(k+1)^d$ such that for every $X \subseteq \mathcal{U}$ of size at most k, X is a minimal hitting set of $\mathcal{F}$ if and only if X is a minimal hitting set of $\mathcal{F}'$.*

We also need the following observation, inspired by a similar observation of Kratsch [27, Lemma 3].

PROPOSITION 5.2. *Let $\mathcal{U}$ be a universe, and let $\mathcal{F}', \mathcal{F}^*, \mathcal{F}$ be sets of subsets of $\mathcal{U}$ such that $\mathcal{F}' \subseteq \mathcal{F}^* \subseteq \mathcal{F}$. Suppose that, for every $X \subseteq \mathcal{U}$ of size at most k, X is a minimal hitting set of $\mathcal{F}$ if and only if X is a minimal hitting set of $\mathcal{F}'$. Then for every $X \subseteq \mathcal{U}$ of size at most k, X is a minimal hitting set of $\mathcal{F}$ if and only if X is a minimal hitting set of $\mathcal{F}^*$.*

Proof. Let $X \subseteq \mathcal{U}$ be of size at most k. Suppose that X is a minimal hitting set of $\mathcal{F}$. Then X is a hitting set of $\mathcal{F}^*$, as $\mathcal{F}^* \subseteq \mathcal{F}$. If a set $X' \subset X$ would be a hitting set of $\mathcal{F}^*$, then X' would also be a hitting set of $\mathcal{F}'$, because $\mathcal{F}' \subseteq \mathcal{F}^*$. However, using the assumption in the proposition statement, X is already a minimal hitting set of $\mathcal{F}'$, a contradiction. Hence, X is a minimal hitting set of $\mathcal{F}^*$.

Suppose that X is a minimal hitting set of $\mathcal{F}^*$. Then X is a hitting set of $\mathcal{F}'$, as $\mathcal{F}' \subseteq \mathcal{F}^*$. Suppose that $X' \subset X$ is a minimal hitting set of $\mathcal{F}'$. This implies that X' is a minimal hitting set of $\mathcal{F}$ by the assumption in the proposition statement. Then X' is also a hitting set of $\mathcal{F}^*$, as $\mathcal{F}^* \subseteq \mathcal{F}$, contradicting the minimality of X. Hence, X is a minimal hitting set of $\mathcal{F}'$. Then the assumption in the proposition statement implies that X is a minimal hitting set of $\mathcal{F}$. ∎

In the remainder, let (G, k) be an instance of (Π_A, Π_B)-RECOGNITION where 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. Throughout, for a graph G', let $\mathcal{F}(G')$ be the set of subsets of $V(G')$ that induce a subgraph of G' isomorphic to a member of $\mathcal{H}$. We observe the following.

PROPOSITION 5.3. *If $B \subseteq V(G)$ is a hitting set of $\mathcal{F}(G)$ and $G[B] \in \Pi_B$, then $(V(G) \setminus B, B)$ is a (Π_A, Π_B)-partition of G.*

Proof. Suppose that a subgraph H of $G - B$ is isomorphic to a member of $\mathcal{H}$. Then $V(H)$ is contained in $\mathcal{F}(G)$. Hence, H contains a vertex of B, a contradiction. Therefore, $G - B \in \Pi_A$. ∎

PROPOSITION 5.4. *If G admits a (Π_A, Π_B)-partition, then G admits a (Π_A, Π_B)-partition (A, B) with $|B|$ minimum such that B is a minimal hitting set of $\mathcal{F}(G)$.*

Proof. Let (A, B) be a (Π_A, Π_B)-partition of G such that $|B|$ is minimum. Clearly, B is a hitting set of $\mathcal{F}(G)$, or $G[A]$ would still contain a subgraph isomorphic to a member of $\mathcal{H}$. Suppose there exists a set $B' \subset B$ such that B' is still a hitting set of $\mathcal{F}(G)$. Note that $G[B'] \in \Pi_B$, because Π_B is hereditary. Hence, Proposition 5.3 implies that $(A \cup (B \setminus B'), B')$ is a (Π_A, Π_B)-partition of G. However, this contradicts the choice of (A, B), particularly the minimality of $|B|$. The proposition follows. ◻

We now describe the single reduction rule of the kernel.

Reduction Rule 5.5. *Apply the algorithm of Lemma* 5.1 *with* $\mathcal{U} = V(G)$, $\mathcal{F} = \mathcal{F}(G)$, *and* k, *and let* $\mathcal{F}'$ *be the resulting set. Let* $T = \bigcup_{F \in \mathcal{F}'} F$ *be the set of vertices contained in* $\mathcal{F}'$, *and let* $R = V(G) \setminus T$. *Remove* R *from* G.

Proof. Let $G' = G[T] = G - R$. We prove that (G, k) is a yes-instance if and only if (G', k) is.

Suppose that (G, k) is a yes-instance, and let (A, B) be a (Π_A, Π_B)-partition of G such that $|B| \leq k$. Since Π_A can be characterized by a collection of forbidden induced subgraphs, it is hereditary. Recall that Π_B is hereditary as well. Hence, $G[A \setminus R] \in \Pi_A$ and $G[B \setminus R] \in \Pi_B$. Therefore, $(A \setminus R, B \setminus R)$ is a (Π_A, Π_B)-partition of G', and (G', k) is a yes-instance.

Suppose that (G', k) is a yes-instance, and let (A', B') be a (Π_A, Π_B)-partition of G' such that $|B'| \leq k$. By Proposition 5.4, we may assume that B' is a minimal hitting set of $\mathcal{F}(G')$. Recall that by Lemma 5.1, for every $X \subseteq \mathcal{U} = V(G)$ of size at most k, X is a minimal hitting set of $\mathcal{F} = \mathcal{F}(G)$ if and only if X is a minimal hitting set of $\mathcal{F}'$. Also note that by the definition of T and G', it follows that $\mathcal{F}' \subseteq \mathcal{F}(G') \subseteq \mathcal{F}(G) = \mathcal{F}$. Combined with Proposition 5.2, all this implies that B' is a (minimal) hitting set of $\mathcal{F}(G)$. But then Proposition 5.3 implies that $(V(G) \setminus B', B') = (A' \cup R, B')$ is a (Π_A, Π_B)-partition of G such that $|B'| \leq k$. Therefore, (G, k) is a yes-instance. ◻

Proof of Theorem 1.2. Let (G, k) be an instance of (Π_A, Π_B)-Recognition, let Π_A be characterized by a collection $\mathcal{H}$ of forbidden induced subgraphs, each of constant size, and let Π_B be hereditary. Apply Reduction Rule 5.5. Since the number of sets in $\mathcal{F}(G)$ is $\mathcal{O}(d|V(G)|^d)$, both constructing $\mathcal{F}(G)$ and the algorithm of Lemma 5.1 take time polynomial in $|V(G)|^d$, d, and k. Moreover, the produced set $\mathcal{F}'$ has size at most $d!\,(k+1)^d$, implying that $|V(G')| \leq d \cdot d!\,(k+1)^d \leq (d+1)!\,(k+1)^d$, where G' is the graph produced by the rule. By the correctness of Reduction Rule 5.5 and the fact that the number of edges in G' is at most quadratic in $|V(G')|$, this is indeed a polynomial kernel. ◻

5.2. Smaller kernels for a restricted setting: Cluster-Π_Δ-Partition. In this subsection, we prove that Cluster-Π_Δ-Partition, parameterized by the size k of B, has a kernel with $\mathcal{O}((\Delta^2+1)k^2)$ vertices. This improves on Theorem 1.2, which implies a kernel with $\mathcal{O}((k+1)^3)$ vertices. Recall that Cluster-Π_Δ-Partition is 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. Throughout, we say that a cluster-Δ partition of G is *valid* if $|B| \leq k$.

The first step of the kernel is to compute a maximal set $\mathcal{P}$ of vertex-disjoint induced P_3s. We call $\mathcal{P}$ a P_3*-packing.* We let $V(\mathcal{P})$ denote the set of vertices of the P_3s in $\mathcal{P}$.

Reduction Rule 5.6. *Let* (G, k) *be an instance of* Cluster-Π_Δ-Partition, *and let* $\mathcal{P}$ *be a* P_3*-packing. If* $|\mathcal{P}| > k$, *then reject.*

Proof. For each P_3, at least one vertex must be in B. Therefore, if $|\mathcal{P}| > k$, then $|B| > k$ for any cluster-Π_Δ partition (A, B) of G. □

Since $\mathcal{P}$ is a maximal set of P_3s, $G - V(\mathcal{P})$ is a cluster graph. The first step of the kernelization is to identify vertices of $V(\mathcal{P})$ that are in B in every valid cluster-Π_Δ partition.

Definition 5.7. *For a vertex $u \in V(\mathcal{P})$, we say that u is* fixed *if either:*

1. *u has neighbors in at least $k+2$ different clusters of $G - V(\mathcal{P})$, or*
2. *there is a cluster C in $G - V(\mathcal{P})$ such that u has (at least) $\Delta + 2$ neighbors and (at least) $\Delta + 2$ nonneighbors in C.*

A fixed vertex u is said to be heavy *if it has neighbors in at least $k+2$ different clusters of $G - V(\mathcal{P})$ (i.e., satisfies condition 1 above); otherwise, u is* nonheavy.

Lemma 5.8. *Let (G, k) be an instance of* Cluster-Π_Δ-Partition, *let $\mathcal{P}$ be a P_3-packing, and let u be a fixed vertex. If G has a valid cluster-Π_Δ partition (A, B), then $u \in B$.*

Proof. Case 1: *u is heavy.* If $u \in A$, then there is at most one cluster C of $G - V(\mathcal{P})$ such that A contains vertices of $N(u) \cup C$. Therefore, B contains vertices of $k+1$ clusters of $G - V(\mathcal{P})$, and thus $|B| > k$.

Case 2: *u is nonheavy.* Since u is fixed, there is a cluster C in $G - V(\mathcal{P})$ such that u has (at least) $\Delta + 2$ neighbors and (at least) $\Delta + 2$ nonneighbors in C. Let $v_1, v_2, \ldots, v_{\Delta+2}$ be $\Delta + 2$ neighbors of u in C, and let $w_1, w_2, \ldots, w_{\Delta+2}$ be $\Delta + 2$ nonneighbors of u in C. Assume, towards a contradiction, that there is a cluster-Π_Δ partition (A, B) with $u \in A$. Since each of $G[\{v_1, v_2, \ldots, v_{\Delta+2}\}]$ and $G[\{w_1, w_2, \ldots, w_{\Delta+2}\}]$ is a clique on $\Delta + 2$ vertices (and hence of degree $\Delta + 1$), A must contain at least one vertex $v_i \in \{v_1, v_2, \ldots, v_{\Delta+2}\}$ and at least one vertex $w_j \in \{w_1, w_2, \ldots, w_{\Delta+2}\}$. But then (u, v_i, w_j) forms an induced P_3 in A. □

Next, we label certain vertices in $V \setminus V(\mathcal{P})$ as *important* using the following scheme.

Labeling Scheme.

(i) For each (fixed) heavy vertex u of $V(\mathcal{P})$, pick $k+2$ (distinct) clusters $C_1, \ldots, C_{k+2}$ in $G - V(\mathcal{P})$ such that, for each $i \in [k+2]$, C_i contains a neighbor v_i of u, and label $v_1, v_2, \ldots, v_{k+2}$ as important.

(iii) For each (fixed) nonheavy vertex u of $V(\mathcal{P})$, pick an arbitrary cluster C of $G - V(\mathcal{P})$ such that u has $\Delta + 2$ neighbors $v_1, v_2, \ldots, v_{\Delta+2}$ and $\Delta + 2$ nonneighbors $w_1, w_2, \ldots, w_{\Delta+2}$ in C, and label $v_1, v_2, \ldots, v_{\Delta+2}, w_1, w_2, \ldots, w_{\Delta+2}$ as important.

(iii) For each nonfixed vertex u of $V(\mathcal{P})$, and each cluster C of $G - V(\mathcal{P})$ containing at least one neighbor of u, label $\min\{\Delta + 2, |N(u) \cap C|\}$ (arbitrary) neighbors of u in C and $\min\{\Delta + 2, |C - N(u)|\}$ (arbitrary) nonneighbors of u in C as important.

Any vertex in $V \setminus V(\mathcal{P})$ that was not labeled in this scheme is called *unimportant*.

Observation 5.9. *If (G, k) is reduced with respect to Reduction Rule* 5.6*, then the number of vertices that are marked as important is $\mathcal{O}((\Delta + 1) \cdot k^2)$.*

Proof. After Reduction Rule 5.6, we have $|V(\mathcal{P})| \leq 3k$. Each heavy vertex in $V(\mathcal{P})$ labels $k + 2$ vertices in $V \setminus V(\mathcal{P})$ as important, according to condition (i) of the labeling scheme. Therefore, the total number of vertices in $V \setminus V(\mathcal{P})$ labeled as important by heavy vertices is $\mathcal{O}(k^2)$. Each fixed nonheavy vertex in $V(\mathcal{P})$ labels

$2\Delta + 4$ vertices in $V \setminus V(\mathcal{P})$ as important, according to condition (ii) of the labeling scheme. Therefore, the total number of vertices in $V \setminus V(\mathcal{P})$ labeled as important by fixed nonheavy vertices is $\mathcal{O}(\Delta \cdot k + k)$. Each nonfixed vertex $v \in V(\mathcal{P})$ is adjacent to at most $k+1$ clusters in $G - V(\mathcal{P})$ (otherwise v would be fixed), and can label at most $2\Delta + 4$ vertices in each adjacent cluster as important (according to condition (iii) of the labeling scheme). Therefore, a nonfixed vertex v labels $\mathcal{O}(\Delta \cdot k + k)$ many vertices in $V \setminus V(\mathcal{P})$ as important. It follows that the at most $3k$ vertices in $V(\mathcal{P})$ label $\mathcal{O}(\Delta \cdot k^2 + k^2) = \mathcal{O}((\Delta + 1) \cdot k^2)$ vertices of $V \setminus V(\mathcal{P})$ as important. ∎

We now present several reduction rules that use the above labeling scheme.

Reduction Rule 5.10. *If there is a cluster C in $G - V(\mathcal{P})$ such that all vertices in C are unimportant, then remove C from G.*

Proof. If G has a valid cluster-Π_Δ partition (A, B), then obviously so does $G - C$. To prove the converse, suppose that (A, B) is a valid cluster-Π_Δ partition of $G - C$. We claim that $(A \cup C, B)$ is a cluster-Π_Δ partition, which obviously satisfies $|B| \le k$, and hence, is valid.

Suppose not. Then there must exist a vertex $u \in A$ that has a neighbor in C. Clearly, $u \in V(\mathcal{P})$ because $G - V(\mathcal{P})$ is a cluster graph containing cluster C and $u \notin C$. Vertex u cannot be fixed; otherwise, since no vertex in C is important, u would remain fixed in $G - C$, and hence, u would not belong to A by Lemma 5.8. Since u is adjacent to C, it follows from condition (iii) of the labeling scheme that at least $\min\{\Delta + 2, |N(u) \cap C|\} > 0$ (since u is adjacent to C) neighbors of u in C are labeled important. This, however, contradicts the assumption of the reduction rule. ∎

Reduction Rule 5.11. *If there is a cluster C in $G - V(\mathcal{P})$ such that C contains (at least) $\Delta + 3$ unimportant vertices, then remove one of these unimportant vertices.*

Proof. Let w be an unimportant vertex in C that is removed by an application of this rule. If G has a valid cluster-Π_Δ partition (A, B), then clearly so does $G - w$. To prove the converse, suppose that $G - w$ has a valid cluster-Π_Δ partition (A, B). We claim that $(A \cup \{w\}, B)$ is a cluster-Π_Δ partition of G, which obviously will be valid.

Since C contains $\Delta + 2$ neighbors $w_1, \ldots, w_{\Delta+2}$ of w that are unimportant and the maximum degree of $G[B]$ is at most Δ, at least one of these vertices, say w_1, belongs to a cluster C' in A. Every vertex in C' that is in $V \setminus V(\mathcal{P})$ must be in C and, hence, is adjacent to w. Now suppose that a vertex $u \in V(\mathcal{P})$ is in C'. We will show that u must be adjacent to w. Suppose, towards a contradiction, that u is not adjacent to w. Since w is unimportant, u cannot be fixed (otherwise, u would be fixed in $G - w$, and would belong to B by Lemma 5.8). Since u is adjacent to $w_1 \in C$, and u is nonfixed, condition (iii) of the labeling scheme applies to u, and, in particular, $\min\{\Delta + 2, |C - N(u)|\}$ nonneighbors of u in C are labeled as important. Since w is a nonneighbor of u in C, and w is unimportant, it follows that there are $\Delta + 2$ nonneighbors of u in C that are different from w and that are labeled important. At least one of these vertices, say x, must be in A. But then (u, w_1, x) forms an induced P_3 in A (note that w_1 is adjacent to x since both of them are in C). This is a contradiction. It follows that each vertex in C' is adjacent to w, and hence, $C' \cup \{w\}$ is a cluster in $A \cup \{w\}$.

To conclude that $G[A \cup \{w\}]$ is a cluster graph, it remains to show that no vertex u that belongs to another cluster $C'' \neq C'$ in $G[A \cup \{w\}]$ is adjacent to w. Suppose not. Then clearly $u \in V(\mathcal{P})$, and by the same arguments as above, u cannot be fixed. Since u is adjacent to $w \in C$, and u is nonfixed, condition (iii) of the labeling scheme applies to u, and, in particular, $\min\{\Delta + 2, |C \cap N(u)|\}$ neighbors of u in C are labeled as important. Since w is unimportant, it follows that there are $\Delta + 2$ neighbors of u in C that are different from w and that are labeled important. One of these neighbors, say x, must be in A and, hence, must belong to the same cluster as both u and w_1 (because $w_1 \in C$). But then this implies that $C' = C''$, contradicting our assumption that u belongs to a different cluster than C'.

It follows that $(A \cup \{w\}, B)$ is a valid cluster-Π_Δ partition of G. ∎

LEMMA 5.12. *Let (G, k) be an instance of* CLUSTER-Π_Δ-PARTITION *that is reduced with respect to the above reduction rules; then G has $\mathcal{O}((\Delta^2 + 1) \cdot k^2)$ vertices.*

Proof. Since (G, k) is reduced with respect to Reduction Rule 5.6, $|V(\mathcal{P})| \leq 3k$. By Observation 5.9, the number of important vertices in $V \setminus V(\mathcal{P})$ is $\mathcal{O}((\Delta+1) \cdot k^2)$. Thus, to show the upper bound on the kernel size, it remains to upper bound the number of unimportant vertices in $V \setminus V(\mathcal{P})$.

To this end, we first upper bound the number of clusters in $G - V(\mathcal{P})$. Since (G, k) is reduced with respect to Reduction Rule 5.10, every cluster in $G - V(\mathcal{P})$ contains at least one important vertex. By Observation 5.9, the number of important vertices in G is $\mathcal{O}((\Delta+1) \cdot k^2)$. Thus, the total number of clusters in $G - V(\mathcal{P})$ is $\mathcal{O}((\Delta+1) \cdot k^2)$.

Now, observe that since (G, k) is reduced with respect to Reduction Rule 5.11, there are at most $\Delta + 3$ unimportant vertices in each cluster and thus $\mathcal{O}((\Delta^2+1) \cdot k^2)$ unimportant vertices overall. ∎

THEOREM 5.13. CLUSTER-Π_Δ-PARTITION*, parameterized by the size k of B, has a polynomial kernel with $\mathcal{O}((\Delta^2 + 1) \cdot k^2)$ vertices that is computable in time $\mathcal{O}(k \cdot (m + n))$, where n and m are the number of vertices and edges, respectively, in the graph.*

Proof. Given an instance (G, k) of CLUSTER-Π_Δ-PARTITION, we start by computing a P_3-packing $\mathcal{P}$. Afterwards, we apply Reduction Rule 5.6–Reduction Rule 5.11. If after the application of these reduction rules the instance (G, k) is not rejected, then these reduction rules result in an equivalent instance (G', k) of CLUSTER-Π_Δ-PARTITION satisfying $|V(G')| = \mathcal{O}((\Delta^2+1) \cdot k^2)$ by Lemma 5.12. Clearly, this implies also that the size of G' is polynomial in k. Therefore, what is left is analyzing the running time taken to apply Reduction Rule 5.6–Reduction Rule 5.11.

First, it is important to observe that each reduction rule is applied exhaustively once, meaning that we apply a particular reduction rule exhaustively, but no more after any of the other reduction rules have been applied. In particular, after applying any of the reduction rules, $G - V(\mathcal{P})$ is still a cluster graph, because the reduction rules only remove vertices. Moreover, the reduction rules leave unchanged the status of each vertex $u \in V(\mathcal{P})$ as (fixed) heavy, (fixed) nonheavy, or nonfixed, because only (edges to) unimportant vertices are removed and the important vertices maintain the status of u. The reduction rules also leave unchanged the label of each vertex in $V \setminus V(\mathcal{P})$ as important or unimportant, for the same reason. Therefore, it suffices to analyze the running time of a single, exhaustive application of each of the reduction rules.

To apply Reduction Rule 5.6, we observe that, as is well known, a P_3 in G can be recognized in $\mathcal{O}(m + n)$ time. (For instance, this can be done by computing the connected components of G and the degree of each vertex in G. We can then identify a connected component that is not a clique, which must exist if a P_3 exists. A P_3 in such a component can then be computed in linear time.) Therefore, $\mathcal{P}$ can be greedily computed in time $\mathcal{O}(k \cdot (m + n))$ (note that if more than k P_3's are identified in G, then the instance can be immediately rejected). It follows from the preceding that

Reduction Rule 5.6 can be applied in $\mathcal{O}(k \cdot (m+n))$ time.

Next, we show that we can classify the vertices in $V(\mathcal{P})$ as fixed heavy, fixed nonheavy, and nonfixed in $\mathcal{O}(m+n)$ time. To do so, we first compute the clusters in $G - V(\mathcal{P})$ and color the vertices of different clusters with different colors, i.e., each vertex in the ith cluster receives color i, for some arbitrary numbering of the clusters. We then iterate through the vertices in $V(\mathcal{P})$, and for each vertex $v \in V(\mathcal{P})$, we iterate through its neighbors in $G - V(\mathcal{P})$. If v has at least $k+2$ neighbors in $G - V(\mathcal{P})$ with different colors (this can be determined in time $\mathcal{O}(\deg(v))$ by sorting the colors of the neighbors of v using Counting Sort), then we define v to be fixed and heavy. For each vertex in $V(\mathcal{P})$ that has not been classified yet, we iterate through its neighbors in $G-V(\mathcal{P})$ and partition its neighbors into subsets, such that all neighbors in the same subset have the same color (belong to the same cluster); for each such subset of neighbors of size $s \geq \Delta + 2$ that belong to a cluster C, we check whether $|C| \geq s + \Delta + 2$, and if it is, we classify v as fixed but nonheavy. All the remaining vertices in $V(\mathcal{P})$ are defined to be nonfixed. Clearly, this whole process can be done in $\mathcal{O}(m+n)$ time.

Afterwards, we label the vertices in $G - V(\mathcal{P})$ as important or unimportant. To do so, for each heavy vertex v in $V(\mathcal{P})$, we iterate through its neighbors in $G - V(\mathcal{P})$ to pick $k+2$ neighbors of distinct colors and label them important. This can be done in time $\mathcal{O}(\deg(v))$, and hence, $\mathcal{O}(m+n)$ time overall. For each fixed nonheavy vertex v in $V(\mathcal{P})$, we iterate through its neighbors to determine a cluster C such that v has $\Delta+2$ neighbors and $\Delta+2$ nonneighbors in C and label those vertices as important. Again, this can be done in time $\mathcal{O}(\deg(v))$, and hence, $\mathcal{O}(m+n)$ time overall. Finally, for each nonfixed vertex v in $V(\mathcal{P})$, we iterate through its neighbors to partition them into subsets of the same color; for each subset of neighbors of the same color that belong to a cluster C, we label $\min\{\Delta+2, |N(u) \cap C|\}$ (arbitrary) neighbors of u in C and $\min\{\Delta+2, |C - N(u)|\}$ (arbitrary) nonneighbors of v in C as important. This can be done in time $\mathcal{O}(\Delta + \deg(v))$, and hence in time $\mathcal{O}(\Delta \cdot (m+n)) = \mathcal{O}(k \cdot (m+n))$ overall.

To apply Reduction Rule 5.10, we go over each cluster C in $G - V(\mathcal{P})$, checking whether it contains any important vertices, and if not, we remove C from G. This can be done in $\mathcal{O}(m+n)$ time.

Finally, to apply Reduction Rule 5.11, we again go over each cluster C in $G - V(\mathcal{P})$, and remove all but $\Delta + 2$ unimportant vertices from C. Again, this can be done in $\mathcal{O}(m+n)$ time. It follows that the kernelization algorithm runs in $\mathcal{O}(k \cdot (m+n))$ time. ∎

6. Conclusion and outlook. As we have seen in this paper, the pushing process is not only useful for finding efficient algorithms for (Π_A, Π_B)-Recognition as demonstrated by Kanj et al. [23], but can also be used to classify when such problems admit polynomial kernels and when they do not. Herein, we focused on the well-motivated case when Π_A is the set of cluster graphs on the first level above triviality, when Π_A is characterized by forbidden induced subgraphs of order at least three. A natural next step is to check to which extent our results carry over to other sets of forbidden subgraphs for Π_A.

The lower bound given in Theorem 1.1 should in a straightforward manner extend to graph classes Π_A that are closed under disjoint union, have neighborhood diversity[3] at most k, and contain cluster graphs. A more challenging avenue is to try to apply

[3]The neighborhood diversity of a graph is the number of different open neighborhoods.

our techniques to related partitioning problems such as Rectangle Stabbing.

Finally, when we parameterized by the size k of one of the parts, we obtained a kernel with $\mathcal{O}(k^d)$ vertices (Theorem 1.2), where d is the largest order of a forbidden subgraph of the other part. Since the techniques used herein are similar to the ones for d-Hitting Set, it is natural to ask whether this upper bound can be improved to $\mathcal{O}(k^{d-1})$.

REFERENCES

[1] F. N. Abu-Khzam, C. Feghali, and H. Müller, *Partitioning a graph into disjoint cliques and a triangle-free graph*, Discrete Appl. Math., 190/191 (2015), pp. 1–12.

[2] D. Achlioptas, *The complexity of G-free colourability*, Discrete Math., 165/166 (1997), pp. 21–30.

[3] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch, *Kernelization lower bounds by cross-composition*, SIAM J. Discrete Math., 28 (2014), pp. 277–305, https://doi.org/10.1137/120880240..

[4] M. Bougeret and P. Ochem, *The complexity of partitioning into disjoint cliques and a triangle-free graph*, Discrete Appl. Math., 217 (2017), pp. 438–445.

[5] H. Broersma, F. V. Fomin, J. Nešetřil, and G. J. Woeginger, *More about subcolorings*, Computing, 69 (2002), pp. 187–203.

[6] S. Bruckner, F. Hüffner, and C. Komusiewicz, *A graph modification approach for finding core-periphery structures in protein interaction networks*, Algorithms Mol. Biol., 10 (2015), 16.

[7] Z. A. Chernyak and A. A. Chernyak, *About recognizing (α, β) classes of polar graphs*, Discrete Math., 62 (1986), pp. 133–138.

[8] R. Churchley and J. Huang, *On the polarity and monopolarity of graphs*, J. Graph Theory, 76 (2014), pp. 138–148.

[9] R. Churchley and J. Huang, *Solving partition problems with colour-bipartitions*, Graphs Combin., 30 (2014), pp. 353–364.

[10] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, New York, 2015.

[11] R. Diestel, *Graph Theory*, 4th ed., Springer, New York, 2012.

[12] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Texts Comput. Sci., Springer, London, 2013.

[13] E. M. Eschen and X. Wang, *Algorithms for unipolar and generalized split graphs*, Discrete Appl. Math., 162 (2014), pp. 195–201.

[14] A. Farrugia, *Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard*, Electron. J. Combin., 11 (2004), R46.

[15] M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette, *On the parameterized complexity of multiple-interval graph problems*, Theoret. Comput. Sci., 410 (2009), pp. 53–61.

[16] J. Fiala, K. Jansen, V. B. Le, and E. Seidel, *Graph subcolorings: Complexity and algorithms*, SIAM J. Discrete Math., 16 (2003), pp. 635–650, https://doi.org/10.1137/S0895480101395245.

[17] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer, New York, 2006.

[18] S. Foldes and P. L. Hammer, *Split graphs*, in Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing (Baton Rouge, LA), Congr. Numer. 19, Utilitas Math., Winnipeg, Manitoba, Canada, 1977, pp. 311–315.

[19] F. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi, *Kernelization: Theory of Parameterized Preprocessing*, Cambridge University Press, Cambridge, UK, 2019.

[20] F. V. Fomin, S. Saurabh, and Y. Villanger, *A polynomial kernel for proper interval vertex deletion*, SIAM J. Discrete Math., 27 (2013), pp. 1964–1976, https://doi.org/10.1137/12089051X.

[21] J. Gimbel and C. Hartman, *Subcolorings and the subchromatic number of a graph*, Discrete Math., 272 (2003), pp. 139–154.

[22] P. Heggernes, D. Kratsch, D. Lokshtanov, V. Raman, and S. Saurabh, *Fixed-parameter algorithms for cochromatic number and disjoint rectangle stabbing via iterative localization*, Inform. Comput., 231 (2013), pp. 109–116.

[23] I. Kanj, C. Komusiewicz, M. Sorge, and E. J. van Leeuwen, *Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs*, J. Comput. System Sci., 92 (2018), pp. 22–47.

[24] I. Kanj, C. Komusiewicz, M. Sorge, and E. J. van Leeuwen, *Solving partition problems almost always requires pushing many vertices around*, in Proceedings of the 26th Annual European Symposium on Algorithms (ESA '18), Leibniz International Proceedings in Informatics (LIPIcs) 112, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Wadern, Germany, 2018, 51.

[25] S. Kolay, F. Panolan, V. Raman, and S. Saurabh, *Parameterized algorithms on perfect graphs for deletion to (r, l)-graphs*, in Proceedings of the 41st MFCS Conference, Leibniz International Proceedings in Informatics (LIPIcs) 58, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Wadern, Germany, 2016, 75.

[26] J. Kratochvíl and I. Schiermeyer, *On the computational complexity of $(\mathcal{O}, \mathcal{P})$-partition problems*, Discuss. Math. Graph Theory, 17 (1997), pp. 253–258.

[27] S. Kratsch, *Polynomial kernelizations for* MIN $f^+\pi_1$ *and* MAX NP, Algorithmica, 63 (2012), pp. 532–550.

[28] V. B. Le and R. Nevries, *Complexity and algorithms for recognizing polar and monopolar graphs*, Theoret. Comput. Sci., 528 (2014), pp. 1–11.

[29] C. McDiarmid and N. Yolov, *Recognition of unipolar and generalised split graphs*, Algorithms, 8 (2015), pp. 46–59.

[30] S. Skiena, *The Algorithm Design Manual*, Springer, New York, 2008.

[31] J. Stacho, *On* 2*-subcolourings of chordal graphs*, in Proceedings of the 8th LATIN Conference on Theoretical Informatics, Lecture Notes in Comput. Sci. 4957, Springer, Berlin, 2008, pp. 544–554.

[32] R. I. Tyshkevich and A. A. Chernyak, *Algorithms for the canonical decomposition of a graph and recognizing polarity*, Izv. Akad. Nauk BSSR Ser. Fiz. Mat. Nauk, 6 (1985), pp. 16–23, in Russian.