

# Kernelization Complexity of Solution Discovery Problems

Mario Grobler ✉ 


University of Bremen, Germany

Stephanie Maaz ✉ 

University of Waterloo, Canada

Amer E. Mouawad ✉ 

American University of Beirut, Lebanon

Naomi Nishimura ✉ 

University of Waterloo, Canada

Vijayaragunathan Ramamoorthi ✉ 

University of Bremen, Germany

Sebastian Siebertz ✉ 

University of Bremen, Germany

---

## Abstract

In the solution discovery variant of a vertex (edge) subset problem  $\Pi$  on graphs, we are given an initial configuration of tokens on the vertices (edges) of an input graph  $G$  together with a budget  $b$ . The question is whether we can transform this configuration into a feasible solution of  $\Pi$  on  $G$  with at most  $b$  modification steps. We consider the token sliding variant of the solution discovery framework, where each modification step consists of sliding a token to an adjacent vertex (edge). The framework of solution discovery was recently introduced by Fellows et al. [ECAI 2023] and for many solution discovery problems the classical as well as the parameterized complexity has been established. In this work, we study the kernelization complexity of the solution discovery variants of VERTEX COVER, INDEPENDENT SET, DOMINATING SET, SHORTEST PATH, MATCHING, and VERTEX CUT with respect to the parameters number of tokens  $k$ , discovery budget  $b$ , as well as structural parameters such as pathwidth.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Graph algorithms; Theory of computation  $\rightarrow$  Fixed parameter tractability; Mathematics of computing  $\rightarrow$  Combinatorics

**Keywords and phrases** solution discovery, kernelization, cut, independent set, vertex cover, dominating set

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2024.36

**Related Version** *Full Version:* <https://arxiv.org/abs/2409.17250>

**Funding** *Naomi Nishimura, Stephanie Maaz:* Research supported by the Natural Sciences and Engineering Research Council of Canada.

*Vijayaragunathan Ramamoorthi:* Funded by the “Mind, Media, Machines” high-profile area at the University of Bremen.

## 1 Introduction

In the realm of optimization, traditional approaches revolve around computing optimal solutions to problem instances from scratch. However, many practical scenarios can be formulated as the construction of a feasible solution from an infeasible starting state. Examples of such scenarios include reactive systems involving human interactions. The inherent dynamics of such a system is likely to lead to an infeasible state. However, computing a



© Mario Grobler, Stephanie Maaz, Amer E. Mouawad, Naomi Nishimura, Vijayaragunathan Ramamoorthi, and Sebastian Siebertz; licensed under Creative Commons License CC-BY 4.0

35th International Symposium on Algorithms and Computation (ISAAC 2024).

Editors: Julián Mestre and Anthony Wirth; Article No. 36; pp. 36:1–36:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

solution from scratch may lead to a solution that may differ arbitrarily from the starting state. The modifications required to reach such a solution from the starting state may be costly, difficult to implement, or sometimes unacceptable.

Let us examine a specific example to illustrate. A set of workers is assigned tasks so that every task is handled by a qualified worker. This scenario corresponds to the classical matching problem in bipartite graphs. Suppose one of the workers is now no longer available (e. g., due to illness); hence, the schedule has to be changed. An optimal new matching could be efficiently recomputed from scratch, but it is desirable to find one that is as close to the original one as possible, so that most of the workers keep working on the task that they were initially assigned.

Such applications can be conveniently modeled using the *solution discovery* framework, which is the central focus of this work. In this framework, rather than simply finding a feasible solution to an instance  $\mathcal{I}$  of a source problem  $\Pi$ , we investigate whether it is possible to transform a given infeasible configuration into a feasible one by applying a limited number of transformation steps. In this work we consider vertex (edge) subset problems  $\Pi$  on graphs, where the *configurations* of the problem are sets of vertices (edges). These configurations are represented by the placement of tokens on the vertices (edges) of the configuration. An atomic *modification step* consists of moving one of the tokens and the question is whether a feasible configuration is reachable after at most  $b$  modification steps. Inspired by the well-established framework of combinatorial reconfiguration [4, 16, 15], commonly allowed modification steps are the addition/removal of a single token, the jumping of a token to an arbitrary vertex/edge, or the slide of a token to an adjacent vertex (edge).

Problems defined in the solution discovery framework are useful and have been appearing in recent literature. Fellows et al. [11] introduced the term *solution discovery*, and along with Grobler et al. [13] initiated the study of the (parameterized) complexity of solution discovery problems for various NP-complete source problems including VERTEX COVER (VC), INDEPENDENT SET (IS), DOMINATING SET (DS), and COLORING (COL) as well as various source problems in P such as SPANNING TREE (ST), SHORTEST PATH (SP), MATCHING (MAT), and VERTEX CUT (VCUT) / EDGE CUT (ECUT).

Fellows et al. [11] and Grobler et al. [13] provided a full classification of polynomial-time solvability vs. NP-completeness of the above problems in all token movement models (token addition/removal, token jumping, and token sliding). For the NP-complete solution discovery problems, they provided a classification of fixed-parameter tractability vs. W[1]-hardness. Recall that a *fixed-parameter tractable algorithm* for a problem  $\Pi$  with respect to a parameter  $p$  is one that solves  $\Pi$  in time  $f(p) \cdot n^{O(1)}$ , where  $n$  is the size of the instance and  $f$  is a computable function dependent solely on  $p$ , while W[1]-hardness provides strong evidence that the problem is likely not fixed-parameter tractable (i. e., does not admit a fixed-parameter tractable algorithm) [9].

A classical result in parameterized complexity theory is that every problem  $\Pi$  that admits a fixed-parameter tractable algorithm necessarily admits a kernelization algorithm as well [5]. A *kernelization algorithm* for a problem  $\Pi$  is a polynomial-time preprocessing algorithm that, given an instance  $x$  of the problem  $\Pi$  with parameter  $p$ , produces a *kernel* – an equivalent instance  $x'$  of the problem  $\Pi$  with a parameter  $p'$ , where both the size of  $x'$  and the parameter  $p'$  are bounded by a computable function depending only on  $p$  [9]. Typically, kernelization algorithms generated using the techniques of Cai et al. [5] yield kernels of exponential (or even worse) size. In contrast, designing problem-specific kernelization algorithms frequently yields more efficiently-sized kernels, often quadratic or even linear with respect to the parameter. Note that once a decidable problem  $\Pi$  with parameter  $p$

admits a kernelization algorithm, it also admits a fixed-parameter tractable algorithm, as a kernelization algorithm always produces a kernel of size that is simply a function of  $p$ . The fixed-parameter tractable solution discovery algorithms of Fellows et al. [11] and Grobler et al. [13] are not based on kernelization algorithms.

Unfortunately, it is unlikely that all fixed-parameter tractable problems admit polynomial kernels. Bodlaender et al. [2, 3] developed the first framework for proving kernel lower bounds and Fortnow and Santhanam [12] showed a connection to the hypothesis  $\text{NP} \not\subseteq \text{coNP/poly}$ . Specifically, for several NP-hard problems, a kernel of polynomial size with respect to a parameter would imply that  $\text{NP} \subseteq \text{coNP/poly}$ , and thus an unlikely collapse of the polynomial hierarchy to its third level [18]. Driven by the practical benefits of kernelization algorithms, we explore the size bounds on kernels for most of the above-mentioned solution discovery problems in the token sliding model, particularly those identified as fixed-parameter tractable in the works of Fellows et al. [11] and Grobler et al. [13].

## 1.1 Results Overview

We focus on the kernelization complexity of solution discovery in the token sliding model for the following source problems: VERTEX COVER, INDEPENDENT SET, DOMINATING SET, SHORTEST PATH, MATCHING, and VERTEX CUT. For a base problem  $\Pi$  we write  $\Pi$ -D for the discovery version in the token sliding model.

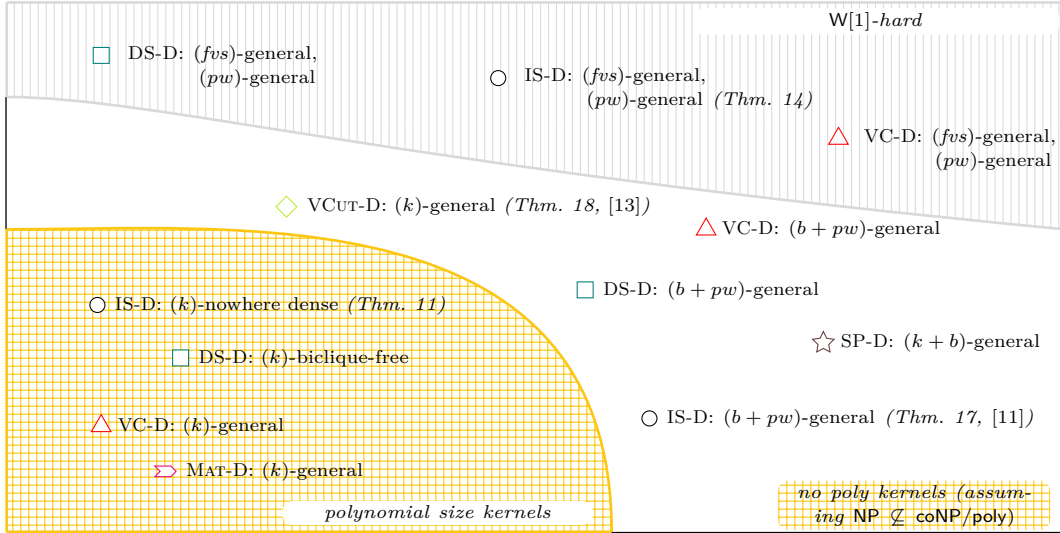
Figure 1 summarizes our results. All graph classes and width parameters appearing in this introduction are defined in the preliminaries. Fellows et al. [11] and Grobler et al. [13] gave fixed-parameter tractable algorithms with respect to the parameter  $k$  for IS-D on nowhere dense graphs, for VC-D, SP-D, MAT-D, and VCUT-D on general graphs and for DS-D on biclique-free graphs.

We show that IS-D, VC-D, DS-D, and MAT-D parameterized by  $k$  admit polynomial size kernels (on the aforementioned classes), while VCUT-D does not admit kernels of size polynomial in  $k$ . For SP-D, we show that the problem does not admit a kernel of polynomial size parameterized by  $k + b$  unless  $\text{NP} \subseteq \text{coNP/poly}$ .

As NP-hardness provides strong evidence that a problem admits no polynomial-time algorithm,  $W[t]$ -hardness (for a positive integer  $t$ ) with respect to a parameter  $p$  provides strong evidence that a problem admits no fixed-parameter tractable algorithm with respect to  $p$ . Fellows et al. [11] proved that VC-D, IS-D, and DS-D are  $W[1]$ -hard with respect to parameter  $b$  on  $d$ -degenerate graphs but provided fixed-parameter tractable algorithms on nowhere dense graphs. They also showed that these problems are slicewise polynomial (XP) with respect to the parameter treewidth and left open the parameterized complexity of these problems with respect to the parameter treewidth alone. We show that these problems remain  $XNLP$ -hard (which implies  $W[t]$ -hardness for every positive integer  $t$ ) for parameter pathwidth (even if given a path decomposition realising the pathwidth), which is greater than or equal to treewidth, and that they admit no polynomial kernels (even if given a path decomposition realising the pathwidth) with respect to the parameter  $b + pw$ , where  $pw$  is the pathwidth of the input graph, unless  $\text{NP} \subseteq \text{coNP/poly}$ .

Finally, we also consider the parameter feedback vertex set number ( $fv_s$ ), which is an upper bound on the treewidth of a graph, but is incomparable to pathwidth. We complement the parameterized complexity classification for the results of Fellows et al. [11] by showing that IS-D, VC-D, and DS-D are  $W[1]$ -hard for the parameter  $fv_s$ .

Several interesting questions remain open. For instance, while their parameterized complexity was determined, the kernelization complexity of COL-D and ECUT-D remains unsettled. Similarly, the kernelization complexity of IS-D and DS-D with respect to parameter  $k$  is unknown on  $d$ -degenerate and semi-ladder-free graphs, respectively, where



■ **Figure 1** A classification of problems into three categories: (yellow, alternatively grid) problems for which we obtain polynomial kernels, (white) those for which polynomial kernels are unlikely, and (grey, alternatively lines) those for which fixed-parameter tractable algorithms are unlikely. Each entry in a category mentions a solution discovery problem, one or more parameters (in parentheses and followed by a dash), and the graph class with respect to which the problem falls into the category. A reference in the parentheses indicates that the fixed-parameter tractability of that problem was established in the cited work.  $pw$  denotes the pathwidth and  $fvs$  denotes the feedback vertex set number of the input graph.

the problems are known to be fixed-parameter tractable. In addition, it remains open whether VCUT-D parameterized by  $k + b$  admits a polynomial kernel or whether MAT-D parameterized by  $b$  admits polynomial kernels on restricted classes of graphs.

## 1.2 Paper Outline

Due to space constraints we cannot present all results in the conference version of the paper. We have chosen to present some results for IS-D and VCUT-D to give an overview of some of our techniques. All other results can be found in the full version. We collect necessary background in Section 2. We show that IS-D has a polynomial size kernel with respect to parameter  $k$  on nowhere dense classes in Section 3. Then, in Section 4 we prove XNLP-hardness with respect to pathwidth. In Section 5, we prove that polynomial kernels with respect to  $k$  are unlikely for VCUT-D.

## 2 Preliminaries

We use the symbol  $\mathbb{N}$  for the set of non-negative integers (including 0),  $\mathbb{Z}$  for the set of all integers, and  $\mathbb{Z}_+$  for the set of positive non-zero integers. For  $k \in \mathbb{N}$ , we define  $[k] = \{1, \dots, k\}$  with the convention that  $[0] = \emptyset$ .

### 2.1 Graphs

We consider finite and simple graphs only. We denote the vertex set and the edge set of a graph  $G$  by  $V(G)$  and  $E(G)$ , respectively, and denote an undirected edge between vertices  $u$  and  $v$  by  $uv$  (or equivalently  $vu$ ) and a directed edge from  $u$  to  $v$  by  $(u, v)$ . We use  $N(v)$  to

denote the set of all neighbors of  $v$  and  $E(v)$  to denote the set of all edges incident with  $v$ . Furthermore, we define the closed neighborhood of  $v$  as  $N[v] = N(v) \cup \{v\}$ . For a set  $X$  of vertices we write  $G[X]$  for the subgraph induced by  $X$ .

A sequence of edges  $e_1 \dots e_\ell$  for some  $\ell \geq 1$  is a (simple) path of length  $\ell$  if every two consecutive edges in the sequence share exactly one endpoint and each other pair of edges share no endpoints. For vertices  $u$  and  $v$ , we denote the length of a shortest path  $e_1 \dots e_\ell$  that connects  $u$  to  $v$  by  $d(u, v)$ , where  $d(v, v) = 0$  for all  $v \in V(G)$ . For a vertex  $v \in V(G)$  and a non-negative integer  $i$ , we denote by  $V(v, i) = \{u \in V(G) \mid d(u, v) = i\}$ .

The complete graph (clique) on  $n$  vertices is denoted by  $K_n$  and a complete bipartite graph (biclique) with parts of size  $m$  and  $n$ , respectively, by  $K_{m,n}$ . We present other relevant graph classes properties in what follows and refer the reader to the textbook by Diestel [7] for an in-depth review of general graph theoretic definitions.

A *tree decomposition* of a graph  $G$  is a pair  $\mathcal{T} = (T, (X_i)_{i \in V(T)})$  where  $T$  is a tree and  $X_i \subseteq V(G)$  for each  $i \in V(T)$ , such that

1.  $\bigcup_{i \in V(T)} X_i = V(G)$ ,
2. for every edge  $uv = e \in E(G)$ , there is an  $i \in V(T)$  such that  $u, v \in X_i$ , and
3. for every  $v \in V(G)$ , the subgraph  $T_v$  of  $T$  induced by  $\{i \in V(T) \mid v \in X_i\}$  is connected, i. e.,  $T_v$  is a tree.

We refer to the vertices of  $T$  as the *nodes* of  $T$ . For a node  $i$ , we say that the corresponding set  $X_i$  is the *bag* of  $i$ . The *width* of the tree decomposition  $(T, (X_i)_{i \in V(T)})$  is  $\max_{i \in V(T)} |X_i| - 1$ . The *treewidth* of  $G$ , denoted  $tw(G)$ , is the smallest width of any tree decomposition of  $G$ . A *path decomposition* of a graph  $G$  is a tree decomposition  $\mathcal{P} = (T, (X_i)_{i \in V(T)})$  in which  $T$  is a path. We represent a path decomposition  $\mathcal{P}$  by the sequence of its bags only. The *pathwidth* of  $G$ , denoted  $pw(G)$ , is the smallest width of any path decomposition of  $G$ .

► **Definition 1.** A class  $\mathcal{C}$  of graphs has *bounded treewidth (bounded pathwidth)* if there exists a constant  $t$  such that all  $G \in \mathcal{C}$  have treewidth (pathwidth) at most  $t$ .

For a graph  $G$ , the *feedback vertex set number* of  $G$  ( $fvs(G)$ ) is the minimum size of a vertex set whose deletion leaves the graph acyclic. We say a graph  $H$  is a *minor* of a graph  $G$ , denoted  $H \preceq G$ , if there exists a mapping that associates each vertex  $v$  of  $H$  with a non-empty connected subgraph  $G_v$  of  $G$  such that  $G_u$  and  $G_v$  are disjoint for  $u \neq v$  and whenever there is an edge between  $u$  and  $v$  in  $H$ , there is an edge between a vertex of  $G_u$  and a vertex of  $G_v$ . The subgraph  $G_v$  is referred to as the *branch set* of  $v$ . We call  $H$  a *depth- $r$  minor* of  $G$ , denoted  $H \preceq_r G$ , if each branch set of the mapping induces a graph of radius at most  $r$ .

► **Definition 2.** A class  $\mathcal{C}$  is *nowhere dense* if there exists a function  $t : \mathbb{N} \rightarrow \mathbb{N}$  such that  $K_{t(r)} \not\preceq_r G$  for all  $r \in \mathbb{N}$  and all  $G \in \mathcal{C}$ .

An  *$r$ -independent set* in a graph  $G$  is a set of vertices  $I$  such that the distance between any two vertices of  $I$  is at least  $r + 1$ . We make use of the fact that nowhere dense classes are uniform quasi-wide, as clarified by the following theorem.

► **Theorem 3** ([14, 17]). Let  $\mathcal{C}$  be a nowhere dense class of graphs. For all  $r \in \mathbb{N}$ , there is a polynomial  $N_r : \mathbb{N} \rightarrow \mathbb{N}$  and a constant  $x_r \in \mathbb{N}$  such that following holds. Let  $G \in \mathcal{C}$  and let  $A \subseteq V(G)$  be a vertex subset of size at least  $N_r(m)$ , for a given  $m \in \mathbb{N}$ . Then there exists a set  $X \subseteq V(G)$  of size  $|X| \leq x_r$  and a set  $B \subseteq A \setminus X$  of size at least  $m$  that is  $r$ -independent in  $G - X$ . Moreover, given  $G$  and  $A$ , such sets  $X$  and  $B$  can be computed in time  $\mathcal{O}(|A| \cdot |E(G)|)$ .

A graph is said to be *d-biclique-free* if it excludes the biclique  $K_{d,d}$  as a subgraph.

► **Definition 4.** A class  $\mathcal{C}$  of graphs is *biclique-free* if there exists a number  $d$  such that all  $G \in \mathcal{C}$  are *d-biclique-free*.

## 2.2 Solution Discovery

A vertex (edge) subset problem  $\Pi$  is a problem defined on graphs such that a solution consists of a subset of vertices (edges) satisfying certain requirements. For a vertex (edge) subset problem  $\Pi$  on an instance with an input graph  $G$ , a *configuration*  $C$  on  $G$  is a subset of its vertices (edges). Alternatively, a configuration can be seen as the placement of tokens on a subset of vertices (edges) in  $G$ . In the *token sliding* model, a configuration  $C'$  can be obtained (in one step) from a configuration  $C$ , written  $C \vdash C'$ , if  $C' = (C \setminus \{y\}) \cup \{x\}$  for elements  $y \in C$  and  $x \notin C$  such that  $x$  and  $y$  are neighbors in  $G$ , that is, if  $x, y \in V(G)$ , then  $xy \in E(G)$ ; and if  $x, y \in E(G)$ , then they share an endpoint. Alternatively, when a token *slides* from a vertex to an adjacent one or from an edge to an incident one, we get  $C \vdash C'$ . A *discovery sequence* of length  $\ell$  in  $G$  is a sequence of configurations  $C_0 C_1 \dots C_\ell$  of  $G$  such that  $C_i \vdash C_{i+1}$  for all  $0 \leq i < \ell$ .

The  $\Pi$ -DISCOVERY problem is defined as follows. We are given a graph  $G$ , a configuration  $S \subseteq V(G)$  (resp.  $S \subseteq E(G)$ ) of size  $k$  (which at this point is not necessarily a solution for  $\Pi$ ), and a budget  $b$  (a non-negative integer). We denote instances of  $\Pi$ -DISCOVERY by  $(G, S, b)$ . The goal is to decide whether there exists a discovery sequence  $C_0 C_1 \dots C_\ell$  in  $G$  for some  $\ell \leq b$  such that  $S = C_0$  and  $C_\ell$  is a solution for  $\Pi$ . When a path decomposition is given as part of the input, the instances are denoted by  $(G, \mathcal{P}_G, S, b)$  to highlight that the path decomposition  $\mathcal{P}_G$  of  $G$  is provided.

## 2.3 Parameterized Complexity and Kernelization

Downey and Fellows [8] developed a framework for parameterized problems which include a parameter  $p$  in their input. A parameterized problem  $\Pi$  has inputs of the form  $(x, p)$  where  $|x| = n$  and  $p \in \mathbb{N}$ . Fixed-parameter tractable problems belong to the complexity class FPT. The class XNLP consists of the parameterized problems that can be solved with a non-deterministic algorithm that uses  $f(p) \cdot \log n$  space and  $f(p) \cdot n^{\mathcal{O}(1)}$  time. The *W-hierarchy* is a collection of parameterized complexity classes  $\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \dots \subseteq \text{XNLP}$  where inclusions are conjectured to be strict.

For parameterized problems  $\Pi$  and  $\Pi'$ , an *FPT-reduction* from  $\Pi$  to  $\Pi'$  is a reduction that given an instance  $(x, p)$  of  $\Pi$  produces  $(x', p')$  of  $\Pi'$  in time  $f(p) \cdot |x|^{\mathcal{O}(1)}$  and such that  $p' \leq g(p)$  where  $f, g$  are computable functions. A *pl-reduction* from  $\Pi$  to  $\Pi'$  is one that additionally computes  $(x', p')$  using  $\mathcal{O}(h(p) + \log |x|)$  working space where  $h$  is a computable function. We write  $\Pi \leq_{\text{FPT}} \Pi'$  (resp.  $\Pi \leq_{\text{pl}} \Pi'$ ) if there is an FPT-reduction (resp. pl-reduction) from  $\Pi$  to  $\Pi'$ . If  $\Pi$  is  $\text{W}[t]$ -hard for a positive integer  $t$  and  $\Pi \leq_{\text{FPT}} \Pi'$ , then  $\Pi'$  is also  $\text{W}[t]$ -hard. If  $\Pi$  is XNLP-hard and  $\Pi \leq_{\text{pl}} \Pi'$ , then  $\Pi'$  is XNLP-hard and, in particular,  $\text{W}[t]$ -hard for all  $t \geq 1$ .

Every problem that is in FPT admits a kernel, although it may be of exponential size or larger. Under the complexity-theoretic assumption that  $\text{NP} \not\subseteq \text{coNP/poly}$ , we can rule out the existence of a polynomial kernel for certain fixed-parameter tractable problems  $\Pi$ . The machinery for such kernel lower bounds heavily relies on composing instances that are equivalent according to a polynomial equivalence relation [6].

► **Definition 5.** An equivalence relation  $\mathcal{R}$  on the set of instances of a problem  $\Pi$  is called a **polynomial equivalence relation** if the following two conditions hold.

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

We can compose equivalent instances in more than one way. We focus here on or-cross-compositions.

► **Definition 6 ([3]).** Let  $\Pi'$  be a problem and let  $\Pi$  be a parameterized problem. We say that  $\Pi$  **or-cross-composes** into  $\Pi'$  if there is a polynomial equivalence relation  $\mathcal{R}$  on the set of instances of  $\Pi$  and an algorithm that, given  $t$  instances (where  $t \in \mathbb{Z}_+$ )  $x_1, x_2, \dots, x_t$  belonging to the same equivalence class of  $\mathcal{R}$ , computes an instance  $(x^*, p^*)$  in time polynomial in  $\sum_{i=1}^t |x_i|$  such that the following properties hold.

1.  $(x^*, p^*) \in \Pi$  if and only if there exists at least one  $i$  such that  $x_i$  is a yes-instance of  $\Pi'$ .
2.  $p^*$  is bounded above by a polynomial in  $\max_{i=1}^t |x_i| + \log t$ .

The inclusion  $\text{NP} \subseteq \text{coNP/poly}$  holds if an NP-hard problem or-cross-composes into a parameterized problem  $\Pi$  having a polynomial kernel. As this inclusion is believed to be false, we will constantly make use of the following theorem to show that the existence of a polynomial kernel is unlikely.

► **Theorem 7 ([3]).** If a problem  $\Pi'$  is NP-hard and  $\Pi'$  or-cross-composes into the parameterized problem  $\Pi$ , then there is no polynomial kernel for  $\Pi$  unless  $\text{NP} \subseteq \text{coNP/poly}$ .

We refer the reader to textbooks [6, 9] for more on parameterized complexity and kernelization.

### 3 IS-D on Nowhere Dense Classes

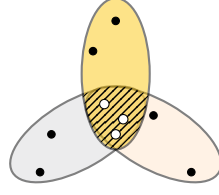
Fellows et al. [11] showed that IS-D is in FPT with respect to parameters  $k$  and  $b$  on nowhere dense classes of graphs. We show in this section that IS-D has a polynomial kernel with respect to parameter  $k$  on the same.

► **Definition 8.** For any instance  $\mathcal{I} = (G, S, b)$  of a  $\Pi$ -DISCOVERY problem for some vertex (resp. edge) selection problem  $\Pi$ , we call a vertex  $v \in V(G) \setminus S$  (resp.  $e \in E(G) \setminus S$ ) **irrelevant with respect to  $s \in S$**  if there exists a configuration  $C_\ell$  such that  $\ell \leq b$ ,  $C_\ell$  is a solution for  $\Pi$ , and the token on  $s$  is not on  $v$  (resp.  $e$ ) in  $C_\ell$ .

The kernelization algorithm for nowhere dense graphs uses Theorem 3, along with other structural properties of the input graph, to form a “sunflower” and find an irrelevant vertex. It then removes from the graph some of the vertices that are irrelevant with respect to every token. A *sunflower* with  $p$  petals and a *core*  $Y$  is a family of sets  $F_1, \dots, F_p$  such that  $F_i \cap F_j = Y$  for all  $i \neq j$ ; the sets  $F_i \setminus Y$  are petals and we require none of them to be empty [10].

► **Lemma 9.** Let  $(G, S, b)$  be an instance of IS-D where  $|S| = k$ , and let  $G'$  be the subgraph of  $G$  induced by the vertices of  $\bigcup_{s \in S, i \in [3k]} V(s, i) \cup S$ . Then  $(G', S, b)$  is equivalent to  $(G, S, b)$ .





■ **Figure 2** An example of a sunflower (with beige, yellow and grey green petals) formed by the closed neighborhoods of the vertices in  $B_j$  of Theorem 11. The vertices in  $B_j$  are 2-independent in  $G - X$  and they have the same closed neighborhood in  $X$  (the white colored vertices).

► **Lemma 10.** *Let  $(G, S, b)$  be an instance of IS-D where  $|S| = k$ , and let  $\mathcal{V} = \{v_1, v_2, \dots, v_t\}$  be a set of vertices of  $G \setminus S$  such that for a given token on a vertex  $s \in S$ ,  $d(s, v_i) = d(s, v_j)$  for  $i \neq j \in [t]$ . If  $\mathcal{A} = \{N[v_1], \dots, N[v_t]\}$  contains a sunflower with  $k + 1$  petals, then any vertex whose closed neighborhood corresponds to one of those petals is irrelevant with respect to  $s$ .*

► **Theorem 11.** *IS-D has a polynomial kernel with respect to parameter  $k$  on nowhere dense graphs.*

**Proof.** Let  $(G, S, b)$  be an instance of IS-D where  $G$  is nowhere dense. Without loss of generality, we assume the graph  $G$  to be connected. For each vertex  $s \in S$  and integer  $i \in [3k]$ , we compute  $V(s, i)$ . We maintain the invariant that we remove from  $V(s, i)$  for each  $s \in S$  and  $i \in [3k]$ , irrelevant vertices with respect to  $s$  (note that a vertex can appear in multiple sets  $V(s, i)$ ).

We remove an irrelevant vertex with respect to a vertex  $s \in S$  from  $V(s, i)$  for an integer  $i \in [3k]$  as follows. If  $|V(s, i)| > N_2(2^{x_2} \cdot (k + 1))$ , where  $N_2$  and  $x_2$  are as per Theorem 3 (here  $V(s, i)$  plays the role of the set  $A$ ), we can compute sets  $X, B \subseteq V(s, i)$  such that  $|X| \leq x_2$ ,  $|B| \geq 2^{x_2} \cdot (k + 1)$  and  $B$  is 2-independent in  $G - X$ . Let  $\mathcal{B}' = \{B'_1, B'_2, \dots\}$  be a family of sets that partitions the vertices in  $B$  such that for any two vertices  $u, v \in B$ ,  $u, v \in B'_j$  if and only if  $N[u] \cap X = N[v] \cap X$ . Since  $|B| \geq 2^{x_2} \cdot (k + 1)$  and  $|X| \leq x_2$ , at least one set  $B_j \in \mathcal{B}'$ , for a specific  $j$ , contains at least  $k + 1$  vertices of  $B$ . All vertices in  $B_j$  have the same neighborhood in  $X$  and they are 2-independent  $G - X$  (i. e., no vertex from outside of  $X$  can be in the closed neighborhood of two vertices in  $B_j$ ); thus their closed neighborhoods form a sunflower with at least  $k + 1$  petals and a core that is contained in  $X$  (Figure 2). By Lemma 10, one vertex of  $B_j$  is irrelevant with respect to  $s$  and can be removed from  $V(s, i)$ . We can repeatedly apply Theorem 3 on the set  $V(s, i)$  until  $|V(s, i)| \leq N_2(2^{x_2} \cdot (k + 1))$ .

We form the kernel  $(G', S, b)$  of the original instance  $(G, S, b)$  as follows. We set  $V(G') = \bigcup_{s \in S, i \in [3k]} V(s, i) \cup S$ . By Lemma 9, any vertex  $v \in V(G)$  such  $d(s, v) > 3k$  for every  $s \in S$  is irrelevant with respect to every  $s \in S$  and not required in the kernel  $(G', S, b)$ . For each vertex  $v \in V(s, i)$ , for  $s \in S$  and  $i \in [3k]$ , we add to  $V(G')$  at most  $i$  vertices that are on the shortest path from  $s$  to  $v$ , if such vertices are not already present in  $V(G')$ .  $G'$  is the subgraph of  $G$  induced by the vertices in  $V(G')$ . By the end of this process,  $|V(G')| \leq k + [9k^3 \cdot N_2(2^{x_2} \cdot (k + 1))]$ , as for each  $s \in S$  and  $i \in [3k]$ ,  $|V(s, i)| \leq N_2(2^{x_2} \cdot (k + 1))$  and for each vertex in the latter sets, we added to  $V(G')$  at most  $3k - 1$  vertices that are on a shortest path from that vertex to the vertex  $s$ .  $(G', S, b)$  is a kernel as only vertices that are irrelevant with respect to every token in  $S$  might not be in  $V(G')$  and all vertices needed to move tokens from vertices in  $S$  towards an independent set using only  $b$  slides are present in  $V(G')$ . ◀



## 4 IS-D for Parameters $b$ and Pathwidth

We now show that IS-D is XNLP-hard with respect to parameter pathwidth. By a small modification of the proof we obtain that IS-D does not have a polynomial kernel with respect to the parameter  $b + pw$ , where  $pw$  is the pathwidth of the input graph, unless  $\text{NP} \subseteq \text{coNP/poly}$ .

### 4.1 The Minimum Maximum Outdegree Problem and Foundational Gadgets

An *orientation* of a graph  $H$  is a mapping  $\lambda : E(H) \rightarrow V(H) \times V(H)$  such that  $\lambda(uv) \in \{(u, v), (v, u)\}$ . Given an undirected weighted graph  $H$ , a path decomposition  $\mathcal{P}_H$  of  $H$  of width  $pw$ , an edge weighting  $\sigma : E(H) \rightarrow \mathbb{Z}_+$  and a positive integer  $r$  (such that all integers are given in unary), the MINIMUM MAXIMUM OUTDEGREE (MMO) asks whether there exists an orientation of  $H$  such that for each  $v \in V(H)$ , the total weight of the edges directed away from  $v$  is at most  $r$ . We use the problem in the reductions that establish the XNLP-hardness of IS-D, VC-D, and DS-D with respect to parameter  $pw$  and the or-cross-compositions that render it unlikely for any of these problems to have a polynomial kernel with respect to parameter  $b + pw$ . Bodlaender et al. [1] showed that MMO is XNLP-complete with respect to pathwidth given a path decomposition realising the pathwidth.

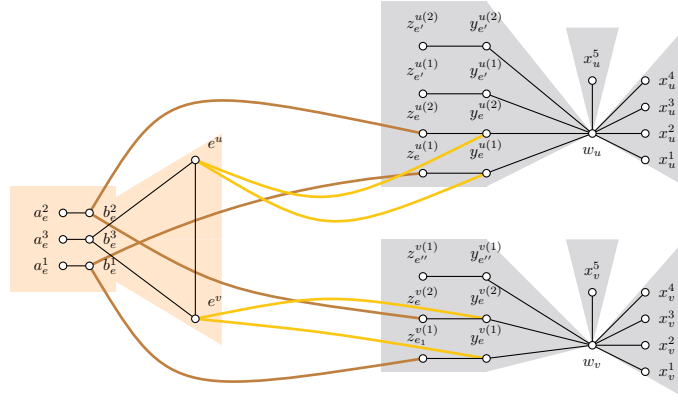
For an instance  $(H, \mathcal{P}_H, \sigma, r)$  of MMO, we define  $\sigma = \sum_{e \in E(H)} \sigma(e)$ ,  $n = |V(H)|$  and  $m = |E(H)|$ . We construct for an instance  $(H, \mathcal{P}_H, \sigma, r)$  of MMO, a graph  $G$  consisting of disjoint subgraphs  $G_e$  for each  $e \in E(H)$  and  $G_v$  for each  $v \in V(H)$ . We refer to the edge-based and vertex-based subgraphs as *MMO-edge-gadgets* and *MMO-vertex-gadgets*, respectively. For an edge  $e \in E(H)$  we refer to  $G_e$  as *MMO-edge- $e$* . Similarly, for a vertex  $v \in V(H)$  we refer to  $G_v$  as *MMO-vertex- $v$* .

**MMO-edge- $e$ .** For an edge  $e = uv \in E(H)$ , an MMO-edge- $e$   $G_e$  contains  $\sigma(e) + 1$  edges with endpoints  $a_e^i$  and  $b_e^i$  for  $i \in [\sigma(e) + 1]$ , and an edge  $e^u e^v$  such that  $b_e^{\sigma(e)+1}$  is adjacent to each of  $e^u$  and  $e^v$ . We define  $A_e = \cup_{i \in [\sigma(e)]} a_e^i$  and  $B_e = \cup_{i \in [\sigma(e)]} b_e^i$ . We refer to the connected component inside  $G_e$  (or any subdivision of  $G_e$ ) containing  $e^u$  and  $e^v$  by  $G_e^{sel}$ .  $\lrcorner$

**MMO-vertex- $v$ .** For a vertex  $v$  in  $V(H)$ , an MMO-vertex- $v$   $G_v$  contains a *representative vertex* of  $v$  denoted by  $w_v$ , adjacent to  $r$  *target vertices* of  $v$  denoted by  $x_v^1, x_v^2, \dots, x_v^r$  and one extra vertex  $x_v^{r+1}$ . Additionally, for each edge  $e \in E(H)$  incident to  $v$ , the MMO-vertex- $v$  contains  $\sigma(e)$  edges with endpoints  $y_e^{v(i)}$  and  $z_e^{v(i)}$  for  $i \in [\sigma(e)]$  such that  $y_e^{v(i)}$  is adjacent to  $w_v$ , the representative vertex of  $v$ . We define  $X_v = \cup_{i \in [r]} x_v^i$ ,  $Y_e^v = \cup_{i \in [\sigma(e)]} y_e^{v(i)}$ ,  $Z_e^v = \cup_{i \in [\sigma(e)]} z_e^{v(i)}$ ,  $Y^v = \cup_{e \in E(H)} Y_e^v$ , and  $Z^v = \cup_{e \in E(H)} Z_e^v$ .  $\lrcorner$

**The Graph  $G$ .** We let  $A = \cup_{e \in E(H)} A_e$ ,  $A^+ = \cup_{e \in E(H)} a_e^{\sigma(e)+1}$ ,  $B = \cup_{e \in E(H)} B_e$ ,  $B^+ = \cup_{e \in E(H)} b_e^{\sigma(e)+1}$ ,  $X = \cup_{v \in V(H)} X_v$ ,  $X^+ = \cup_{v \in V(H)} x_v^{r+1}$ ,  $Y = \cup_{v \in V(H)} Y^v$ , and  $Z = \cup_{v \in V(H)} Z^v$ . We form  $G$  by connecting its MMO-edge-gadget vertices to its MMO-vertex-gadget vertices as follows. For a vertex  $v \in V(H)$  and edge  $e \in E(H)$  incident to  $v$ , we connect each vertex of  $B_e$  to a corresponding distinct vertex in  $Z_e^v$  (in other words, each  $b_e^i$  to  $z_e^{v(i)}$  for  $i \in [\sigma(e)]$ ). Similarly, we connect  $e^v$  to each vertex of  $Y_e^v$  (see Figure 3 for an example).  $\lrcorner$

Our reductions must use at most  $\mathcal{O}(h(pw) + \log |x|)$  working space, for an input instance of size  $|x|$  and parameter  $pw$ , and a computable function  $h$ . We show that our reductions/-compositions can be performed on a *log-space transducer* and are pl-reductions. A log-space



■ **Figure 3** Edges from one MMO-edge- $e$ , for an edge  $e = uv$  for a graph  $H$ , edge weight function  $\sigma$ , and integer  $r$  of an MMO instance, to the MMO-vertex- $u$  and MMO-vertex- $v$  subgraphs in  $G$ . Brown is used for edges between vertices in  $B$  and  $Z$  and yellow is used for edges between vertices in  $\{e^u, e^v\}$  and  $Y$ .  $\sigma(e) = 2$  and  $r = 4$ .

transducer is a type of Turing machine with a read-only input tape, a read/write work tape of logarithmic size and a write-only, write-once output tape. For the graph  $G$ , we show the following lemma.

▶ **Lemma 12.** *Let  $(H, \mathcal{P}_H, w, r)$  be an instance of MMO. Then, there exists a log-space transducer that transforms a path decomposition of  $H$  to one of  $G$  with width at most  $\text{pw}(H) + 6$ . Thus,  $\text{pw}(G) \leq \text{pw}(H) + 6$ .*

▶ **Corollary 13.** *Given an MMO instance  $(H, \mathcal{P}_H, \sigma, r)$ , one can build a log-space transducer that outputs a path decomposition of  $G$  with width at most  $\text{pw}(H) + 6$ , along with a representation of the graph, any subset of its vertices, and an integer with at most a polynomial (in the input size) number of bits.*

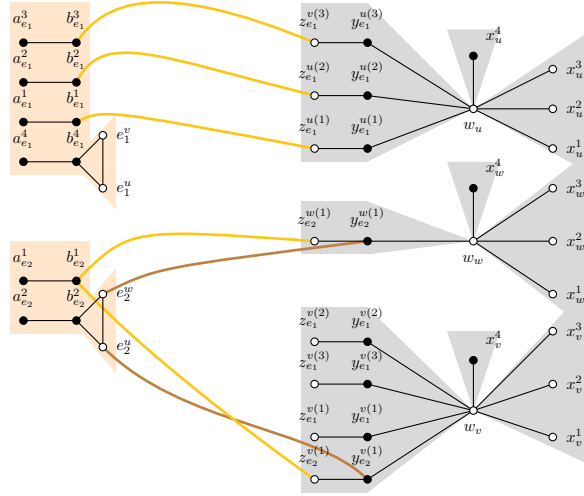
## 4.2 Lower Bound Proofs

▶ **Theorem 14.** *IS-D is XNLP-hard with respect to parameter pathwidth.*

**Proof.** We present an fpt-reduction from MMO. Let  $(H, \mathcal{P}_H, \sigma, r)$  be an instance of MMO where  $H$  is a bounded pathwidth graph,  $|V(H)| = n$ ,  $|E(H)| = m$ ,  $\sigma : E(H) \rightarrow \mathbb{Z}_+$  such that  $\sum_{e \in E(H)} \sigma(e) = \sigma$  and  $r \in \mathbb{Z}_+$  (integers are given in unary). We construct an instance  $(G, \mathcal{P}_G, S, b)$  of IS-D where  $G$  is exactly as described in Section 4.1. See Figure 4. We set  $S = A \cup A^+ \cup B \cup B^+ \cup Y \cup X^+$  and  $b = m + 3\sigma$ . Given that all integers are given in unary, the construction of the graph  $G$ , or its path decomposition (as described in Lemma 12), and as a consequence the reduction, take time polynomial in the size of the input instance. Additionally, by Corollary 13, this reduction is a pl-reduction. We claim that  $(H, \mathcal{P}_H, \sigma, r)$  is a yes-instance of MMO if and only if  $(G, \mathcal{P}_G, S, b)$  is a yes-instance of IS-D.

▷ **Claim 15.** If  $(H, \mathcal{P}_H, \sigma, r)$  is a yes-instance of MMO, then  $(G, \mathcal{P}_G, S, b)$  is a yes-instance of IS-D.

**Proof.** Let  $\lambda : E(H) \rightarrow V(H) \times V(H)$  be an orientation of the graph  $H$  such that for each  $v \in V(H)$ , the total weight of the edges directed out of  $v$  is at most  $r$ . In  $(H, \mathcal{P}_H, \sigma, r)$ , the vertices in  $A$  and  $B$  contain tokens. The same applies for the vertices in  $A^+$  and  $B^+$ . To fix that, for each edge  $e \in E(H)$  such that  $\lambda(e) = (v, u)$ :



■ **Figure 4** Parts of the graph  $G$  constructed by the reduction of Theorem 14 given an instance  $(H, \mathcal{P}_H, \sigma, r)$ , where  $H$  has three vertices  $u, v$  and  $w$ , and two edges  $e_1 = uv$  and  $e_2 = uw$ , and  $r = 3$ . Additionally,  $\sigma(e_1) = 3$  and  $\sigma(e_2) = 1$ . For clarity, the edges between the vertices in  $B_{e_1}$  and  $Z_{e_1}^u$  are missing. The same applies for the edges between  $e_1^v$  and the vertices of  $Y_{e_1}^v$  and the edges between  $e_1^u$  and the vertices of  $Y_{e_1}^u$ . Brown and yellow edges are used to highlight the different types of edges used to connect the subgraphs  $G_{e_1}, G_{e_2}, G_u, G_v$  and  $G_w$  of  $G$ , vertices in black are in  $S$  and those in white are not.

1. we slide, for each  $i \in [\sigma(e)]$ , the token on  $b_e^i$  to  $z_e^{v(i)}$  (this consumes  $\sigma(e)$  slides),
2. we move, for each  $i \in [\sigma(e)]$ , the token on  $y_e^{v(i)}$  to any free vertex of  $X_v$  (this consumes  $2\sigma(e)$  slides),
3. we slide the token on  $b_e^{\sigma(e)+1}$  to  $e^v$  (this consumes 1 slide).

This constitutes  $m + 3\sigma$  slides and we get an independent set in  $G$ . Step 2 above is possible (i. e., a token-free vertex exists in  $X^v$ ) since  $\lambda$  is an orientation of the graph  $H$  such that for each  $v \in V(H)$ , the total weight of the edges directed out of  $v$  is at most  $r$ . Step 3 is possible for each edge  $e \in E(H)$  since in Step 2 all tokens were removed from the vertices in  $Y_e^v$ .  $\triangleleft$

▷ **Claim 16.** If  $(G, \mathcal{P}_G, S, b)$  is a yes-instance of IS-D, then  $(H, \mathcal{P}_H, \sigma, r)$  is a yes-instance of MMO.

*Proof.* The minimum number of slides used inside any induced subgraph  $G_e$  for an edge  $uv = e \in E(H)$  is one and it can only be achieved by sliding the token on  $b_e^{\sigma(e)+1}$  to one of either  $e^u$  or  $e^v$ . Thus, at least  $m$  slides are required inside the MMO-edge-gadgets and the budget remaining is  $3\sigma$ . Additionally, each token on a vertex  $b_e^i$  in  $B_e$ , for an edge  $uv = e \in E(H)$  and an integer  $i \in [\sigma(e)]$  must slide to either  $z_e^{u(i)}$  or  $z_e^{v(i)}$ , consuming  $\sigma$  slides. Since a solution that moves the token on  $a_e^{\sigma(e)+1}$  but not the token on  $b_e^{\sigma(e)+1}$  is not minimal, we can safely assume that the described  $m + \sigma$  slides are executed in any minimal solution.

In the same solutions, each token on a vertex  $z_e^{u(i)}$  for an edge  $uv = e \in E(H)$  and an integer  $i \in [\sigma(e)]$  requires the token on  $y_e^{u(i)}$  to slide to either  $e^u$  or  $w_u$ , utilizing as a result  $\sigma$  other slides. A token that slides from  $y_e^{u(i)}$  to the vertex  $w_u$  must slide again at least once, since any independent set that is achieved through the minimal number of slides would never require the sliding of the tokens on the vertices in  $X^+$  (the token that moves to the vertex  $w_u$

can be moved, using one less slide, to the vertex the token on  $x_u^{r+1}$  moves to). Since  $G_e^{sel}$  can contain at most 2 tokens, a token on  $y_e^{u(i)}$  that slides to the vertex  $e^u$  must either slide again at least once to a vertex, denoted  $y_e^{u(i_1)}$  (for an integer  $i_1 \in [\sigma(e)]$ ) in  $Y_e^u$ , or require another token on a vertex in  $G_e^{sel}$  to slide at least once to either a vertex, denoted  $y_e^{u(i_2)}$  (for an integer  $i_2 \in [\sigma(e)]$ ) in  $Y_e^u$ , or a vertex, denoted  $y_e^{v(i_2)}$  (for an integer  $i_2 \in [\sigma(e)]$ ) in  $Y_e^v$ , while the token initially on  $y_e^{u(i)}$  stays on  $e^u$ . Given that at most  $\sigma$  slides remain in any minimal solution, and that each of the  $\sigma$  tokens initially on vertices in  $Y$  that moved to either vertices of the form  $e_1^{u_1}$  or  $w_{u_1}$ , for an edge  $e_1 \in E(H)$  incident to a vertex  $u_1 \in V(H)$ , uses or requires at least one additional slide, each one such token can use or require exactly one additional slide. If the token on  $y_e^{u(i)}$  slides to  $w_u$ , then either in exactly one more slide it can move to a free vertex in  $X_u$ , or it can slide back to a vertex, denoted  $y_{e_2}^{u(i_3)}$  (for an edge  $e_2$  adjacent to  $u$  in  $H$  and an integer  $i_3 \in [\sigma(e_1)]$ ) in  $Y^u$ . However, either  $y_{e_2}^{u(i_3)}$  (resp.  $y_e^{u(i_1)}$ ,  $y_e^{u(i_2)}$ , or  $y_e^{v(i_2)}$ ) or its adjacent vertex, denoted  $z_{e_2}^{u(i_3)}$  in  $Z^u$  (resp.  $z_e^{u(i_1)}$  in  $Z_e^u$ ,  $z_e^{u(i_2)}$  in  $Z_e^u$ , or  $z_e^{v(i_2)}$  in  $Z_e^v$ ), contains a token, thus requiring at least one other additional slide, which is impossible. As a result, it can only be the case that a token on  $y_e^{u(i)}$  slides to  $w_u$  and then in exactly one more slide it moves to a free vertex in  $X_u$ .

For any edge  $uv = e \in E(H)$ , if  $e^v \in C_\ell$  (resp.  $e^u \in C_\ell$ ), then no vertex of  $Y_e^v$  (resp.  $Y_e^u$ ) appears in  $C_\ell$  and the tokens on the vertices of  $Y_e^v$  (resp.  $Y_e^u$ ) have been moved to some of the free vertices of  $X_v$  (resp.  $X_u$ ). Given the latter, we produce an orientation  $\lambda$  to  $H$ , where  $\lambda(e) = (v, u)$  (resp.  $\lambda(e) = (u, v)$ ) if  $e^v \in C_\ell$  (resp.  $e^u \in C_\ell$ ). Since  $|X_v| = |X_u| \leq r$ ,  $\lambda$  is such that the total weight directed out of any vertex  $v \in V(H)$  is at most  $r$ .  $\square$

This concludes the proof of the theorem.  $\blacktriangleleft$

We compose multiple MMO instances utilizing the construction presented in Theorem 14, and show the following.

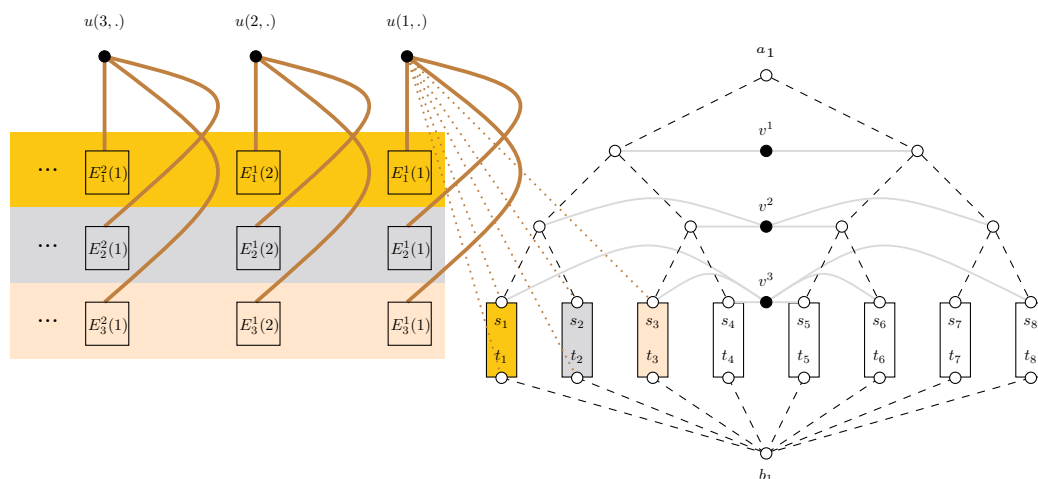
► **Theorem 17.** *IS-D does not admit a polynomial kernel with respect to  $b + pw$ , where  $pw$  denotes the pathwidth of the input graphs, unless  $NP \subseteq \text{coNP/poly}$ .*

## 5 VCut-D for Parameter $k$

Grobler et al. [13] showed that V<sub>CUT</sub>-D is W[1]-hard with respect to parameter  $b$  on 2-degenerate bipartite graphs but is in FPT with respect to the parameter  $k$  on general graphs. We show that the problem admits no polynomial kernels unless  $NP \subseteq \text{coNP/poly}$ . We denote an instance of V<sub>CUT</sub>-D by  $(G, S, b, a_1, b_1)$  to emphasize that the solution must be a vertex cut between the vertices  $a_1$  and  $b_1$  in  $V(G)$ .

Given a graph  $H$  and an edge coloring  $\phi : E(H) \rightarrow [c]$ , we say  $\phi$  is proper if, for all distinct edges  $e, e_1 \in E(H)$ ,  $\phi(e) \neq \phi(e_1)$  whenever  $e$  and  $e_1$  share a vertex. We form our or-cross-composition from the RAINBOW MATCHING problem, which is NP-complete even on properly colored 2-regular graphs and where every  $i \in [c]$  is used exactly twice in the coloring [15]. Given a graph  $H$ , a proper edge coloring  $\phi$  and an integer  $\kappa$ , the RAINBOW MATCHING problem asks whether  $(H, \phi, \kappa)$  has a rainbow matching of size  $\kappa$ , i.e., a matching whose edges have distinct colors, with at least  $\kappa$  edges.

► **Theorem 18.** *There exists an or-cross-composition from RAINBOW MATCHING into V<sub>CUT</sub>-D where the parameter is the number of tokens,  $k$ . Consequently, V<sub>CUT</sub>-D does not admit a polynomial kernel with respect to  $k$ , unless  $NP \subseteq \text{coNP/poly}$ .*



■ **Figure 5** An illustration of the graph  $G$  formed as per the composition of Theorem 18 given input instances  $(H_r, \phi_r, \kappa_r)$  for  $r \in [8]$ , where  $\kappa_r = \kappa \geq 3$ . For clarity, each graph  $G_r$  for  $r \in [8]$  was replaced by a rectangle incident to two vertices  $s_r$  and  $t_r$  of  $G_r$ . Grey edges are used to illustrate how the vertices  $v^d$  for  $d \in [3]$  connect to the vertices of  $\mathcal{T}$ . Dashed lines represent paths of length  $m^3 + \log t$  between the vertices and thick edges are used to represent that a vertex is adjacent to all vertices in a set of vertices. The yellow, grey, and beige rectangular areas on the left provide a zoomed-in view of some of the content of  $G_1$ ,  $G_2$ , and  $G_3$ , respectively. Particularly, they show the sets of vertices  $E_1^1(1)$ ,  $E_1^1(2)$ ,  $E_1^2(1)$ ,  $E_2^1(1)$ ,  $E_2^1(2)$ ,  $E_2^2(1)$ ,  $E_3^1(1)$ ,  $E_3^1(2)$ , and  $E_3^2(1)$ . For clarity, not all (dotted line) edges between vertices of the form  $u(i, j)$  for  $i \in [2(\kappa - 1)]$  and  $j \in [m - 1]$ , and both vertices  $s_r$  and  $t_r$  for  $r \in [8]$  are shown.

**Proof.** By choosing an appropriate polynomial equivalence relation  $\mathcal{R}$ , we may assume that we are given a family of  $t$  RAINBOW MATCHING instances  $(H_r, \phi_r, \kappa_r)$ , where  $H_r$  is a 2-regular graph,  $|V(H_r)| = n$ ,  $|E(H_r)| = m$ ,  $\kappa_r = \kappa \in \mathbb{N}$ , and  $\phi_r : E(H_r) \rightarrow [c]$  is a mapping that properly colors  $H_r$  and in which every  $i \in [c]$  is used exactly twice. We may duplicate some input instances so that  $t = 2^s$  for some integer  $s$ . Note that this step at most doubles the number of input instances. The construction of the instance  $(G, S, b, a_1, b_1)$  of VCUT-D is twofold.

For each instance  $(H_r, \phi_r, \kappa_r)$ , we create  $G_r$ , formed of two vertices,  $s_r$  and  $t_r$ , as well as  $\kappa - 1$  sets  $\{E_r^1, \dots, E_r^{\kappa-1}\}$  of  $2m + 2$  vertices each. A set  $E_r^p$  for  $p \in [\kappa - 1]$  contains  $2m$  vertices, denoted *edge-vertices*, that represent the edges in  $H_r$  twice and two other vertices which are denoted by  $s_r^p$  and  $t_r^p$  (see Figure 6). We denote the edge-vertices in a set  $E_r^p$  as  $v_{e_h}^{p,r}(1)$  ( $v_{e_h}^{p,r}(2)$ ) to refer to the first (second) vertex representing the same edge  $e_h$  of  $E(H_r)$  in  $E_r^p$ . We denote by  $E_r^p(1)$  the set of all vertices  $v_{e_h}^{p,r}(1)$ , and by  $E_r^p(2)$  the set of all vertices  $v_{e_h}^{p,r}(2)$ . In  $G_r$ , we connect through paths of length  $m^3 + \log t$ :

- $s_r$  to each of  $s_r^p$  for  $p \in [\kappa - 1]$  and  $t_r$  to each of  $t_r^p$  for  $p \in [\kappa - 1]$ ,
- $s_r^p$  to all vertices  $v_{e_h}^{p,r}(1)$  and  $t_r^p$  to all vertices  $v_{e_h}^{p,r}(2)$  for each  $e_h \in E(H_r)$  and each  $p \in [\kappa - 1]$ ,
- all vertices  $v_{e_h}^{p,r}(1)$  and  $v_{e_g}^{q,r}(2)$  such that  $\phi_r(e_h) = \phi_r(e_g)$  for each  $p \leq q \in [\kappa - 1]$ ,
- $v_{e_h}^{p,r}(1)$  and  $v_{e_g}^{q,r}(2)$ , for each  $p \leq q \in [\kappa - 1]$ , whenever  $e_h$  and  $e_g$  are incident in  $H_r$ ,
- $v_{e_h}^{p,r}(2)$  and  $v_{e_g}^{q,r}(1)$ , for each  $p \in [\kappa - 2]$ ,  $q = p + 1$ , whenever  $e_h \neq e_g$ .

We form  $G$  of all  $G_r$  for  $r \in [t]$  as follows (see Figure 5). We create two global vertices  $a_1$  and  $b_1$  such that  $b_1$  is connected through paths of length  $m^3 + \log t$  to  $t_r$  for  $r \in [t]$ . Additionally, we create a binary tree  $\mathcal{T}$  rooted at  $a_1$ , with  $\log t + 1$  levels, and whose leaves

## 36:14 Kernelization Complexity of Solution Discovery

constitute  $s_r$  for  $r \in [t]$ . For each depth  $d$  of  $\mathcal{T}$  for  $d \in \{1, \dots, \log t\}$ , we create a vertex  $v^d$  that contains a token and is connected through a single edge to each vertex of  $\mathcal{T}$  that is at depth  $d$ . The edges of  $\mathcal{T}$  are all replaced by paths of length  $m^3 + \log t$ . Finally, we create  $2(\kappa - 1)$  sets  $\{M_1, \dots, M_{2(\kappa-1)}\}$ , of  $m-1$  edges each. We connect each edge  $e^{(i,j)} \in M_i$  for  $i \in [2(\kappa-1)]$  and  $j \in [m-1]$ , from one of its endpoints, denoted  $u^{(i,j)}$ , to each vertex  $v_{e_h}^{\lceil i/2 \rceil, r}(1)$  for each  $r \in [t]$  if  $i$  is odd, and to each vertex  $v_{e_h}^{\lceil i/2 \rceil, r}(2)$  for each  $r \in [t]$  if  $i$  is even. Additionally, we connect through paths of length  $m^3 + \log t$ , each  $s_r$  and  $t_r$  for  $r \in [t]$  to all of  $u^{(i,j)}$  for  $i \in [2(\kappa-1)]$  and  $j \in [m-1]$ . All vertices in the sets  $\{M_1, \dots, M_{2(\kappa-1)}\}$  contain tokens. Setting  $b = \log t + 2(2\kappa - 2) \cdot (m - 1)$  finalizes the construction of  $(G, S, b, a_1, b_1)$ . Since we perform only a polynomial number of operations per instance as well as some polynomial in  $t$  other operations while creating the tree  $\mathcal{T}$  and connecting some vertices, the reduction is polynomial in  $\sum_{i=1}^t |x_i|$ . Additionally,  $k$  is  $O(m^2 + \log t)$  since  $\kappa \leq m$ .

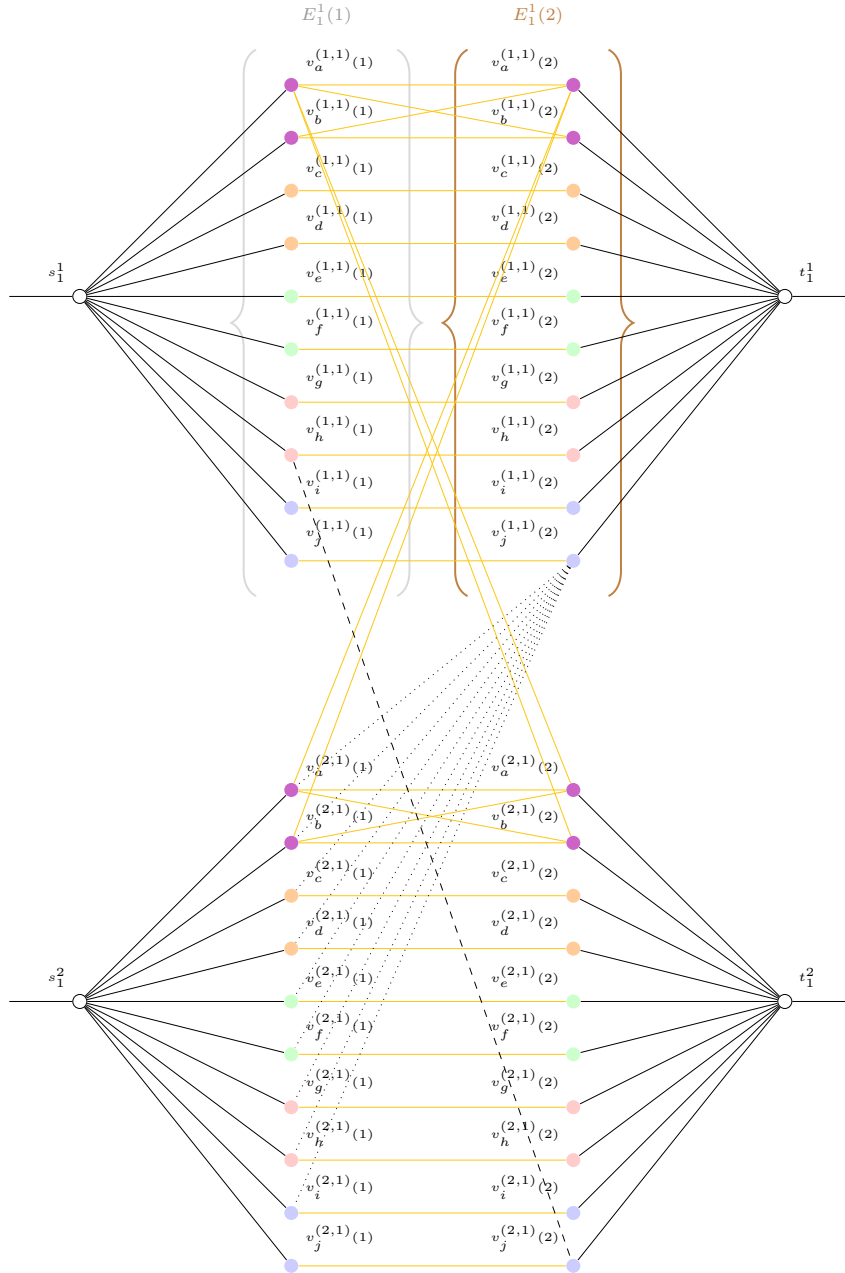
▷ **Claim 19.** If for some  $\tau \in [t]$ ,  $(H_\tau, \phi_\tau, \kappa_\tau)$  is a yes-instance of RAINBOW MATCHING, then the constructed instance  $(G, S, b, a_1, b_1)$  is a yes-instance of VCUT-D.

*Proof.* Let  $\mathcal{M}_\tau$  be a solution to the instance  $(H_\tau, \phi_\tau, \kappa_\tau)$ .  $\mathcal{M}_\tau \subseteq E(H_\tau)$  forms a matching in  $H_\tau$  such that  $\phi_\tau(e_h) \neq \phi_\tau(e_g)$ , for all  $e_h, e_g \in \mathcal{M}_\tau$ . We apply the following slides in  $(G, S, b, a_1, b_1)$  to disconnect  $a_1$  from  $b_1$ . First, we choose one edge  $e_h$  of  $\mathcal{M}_\tau$  and using  $m - 1$  slides, we slide the tokens on  $u^{(1,j)}$  for  $j \in [m - 1]$  onto all vertices in  $E_\tau^1(1)$  except  $v_{e_h}^{1,\tau}(1)$ . Then, using  $(2\kappa - 1) \cdot (m - 1)$  slides, for each  $i \in [\kappa - 1]$ , we choose one other edge  $e_s \in \mathcal{M}_\tau$  and slide the tokens on  $u^{(2i,j)}$  and  $u^{(2i+1,j)}$  (when applicable) for  $j \in [m - 1]$  onto all vertices in  $E_\tau^i(2)$  and  $E_\tau^{i+1}(1)$  except  $v_{e_s}^{i,\tau}(2)$  and  $v_{e_s}^{i+1,\tau}(1)$ , respectively. We slide onto  $u^{(i,j)}$  for all  $i \in [2(\kappa - 1)]$  and  $j \in [m - 1]$  the tokens adjacent to the latter vertices, on the edges in  $\{M_1, \dots, M_{2(\kappa-1)}\}$ , using  $(2\kappa - 2) \cdot (m - 1)$  slides. Finally, in  $\mathcal{T}$ , we use the tokens on the vertices  $v^d$  for  $d \in \{1, \dots, \log t\}$ , to disconnect all paths from the root  $a_1$  to all of  $s_r$  for  $r \in [t] - \{\tau\}$ , using one slide per token. This ensures that, through at most  $\log t$  slides, all paths from  $a_1$  to  $b_1$  go through only both  $s_\tau$  and  $t_\tau$ . Following the described steps, we have executed a total of  $b$  slides. To see that  $a_1$  and  $b_1$  are now disconnected, note that after the slides of the tokens on  $v^d$  for  $d \in \{1, \dots, \log t\}$  are performed, all paths from  $a_1$  to  $b_1$  in  $G$  go through  $s_\tau$  and  $t_\tau$ . Thus it suffices to argue that the remaining  $2(2\kappa - 2) \cdot (m - 1)$  slides disconnect  $s_\tau$  and  $t_\tau$ . First, if this is not the case, then no path between  $s_\tau$  and  $t_\tau$  goes through any  $u^{(i,j)}$  for all  $i \in [2(\kappa - 1)]$  and  $j \in [m - 1]$  since the tokens that left those vertices have been replaced. Also, the last four vertices on any path between  $s_\tau$  and  $t_\tau$  must be  $v_{e_h}^{p,\tau}(1)$  for some  $p \in [\kappa - 1]$  and some  $e_h \in E(H_\tau)$ ,  $v_{e_g}^{q,\tau}(2)$  for some  $q \in \{p, \dots, \kappa - 1\}$  and some  $e_g \in E(H_\tau)$ ,  $t_\tau^q$  and  $t_\tau$ . However, by construction, there exists no paths between all vertices  $v_{e_h}^{p,\tau}(1)$  and  $v_{e_g}^{q,\tau}(2)$  for each  $p \leq q \in [\kappa - 1]$ , such that  $\phi_\tau(e_h) \neq \phi_\tau(e_g)$  and  $e_h$  and  $e_g$  are non-adjacent. Thus, given our choice of the free vertices remaining in  $E_\tau^p(\cdot)$  for all  $p \in [\kappa - 1]$ , no path exists between  $s_\tau$  from  $t_\tau$  and therefore between  $a_1$  and  $b_1$ . ◁

▷ **Claim 20.** If  $(G, S, b, a_1, b_1)$  is a yes-instance of VCUT-D, then there exists an integer  $\tau \in [t]$  for which  $(H_\tau, \phi_\tau, \kappa_\tau)$  is a yes-instance of RAINBOW MATCHING.

*Proof.* Assume  $C_\ell$  for  $\ell \leq b$ , is a solution to  $(G, S, b, a_1, b_1)$  that is reached with only  $2(2\kappa - 2) \cdot (m - 1) + \log t$  slides and disconnects  $a_1$  from  $b_1$ , then any token that slides in  $G$  slides at most once, given that everything except:

- for  $d \in \{1, \dots, \log t\}$ , the vertex  $v^d$  and each vertex of  $\mathcal{T}$  that is at level  $d$ ,
- $u^{(i,j)}$  for  $i \in [2(\kappa - 1)]$  and  $j \in [m - 1]$ , to each vertex  $v_{e_h}^{\lceil i/2 \rceil, r}(1)$  for each  $r \in [t]$  if  $i$  is odd, and to each vertex  $v_{e_h}^{\lceil i/2 \rceil, r}(2)$  for each  $r \in [t]$  if  $i$  is even,
- the endpoints of each edge  $e^{(i,j)} \in M_i$  for  $i \in [2(\kappa - 1)]$  and  $j \in [m - 1]$ ,



■ **Figure 6** An illustration of  $E_1^1$ ,  $E_1^2$ ,  $s_1^1$ ,  $t_1^1$ ,  $s_1^2$ , and  $t_1^2$  of  $G_1$  of the or-cross-composition of Theorem 18. In  $H_1$ , the vertices are  $a, b, c, d, e, f, g, h, i$ , and  $j$ . For simplification purposes, the figure illustrates the types of edges but does not contain all edges between the illustrated vertices. Length  $m^3 + \log t$  paths are represented by the edges (regular, dotted or dashed). Vertices in grey brackets are in  $E_1^1(1)$  and those in brown brackets are in  $E_1^1(2)$ . Yellow edges are between vertices representing edges of the same color in  $H^1$  and dotted ones between all  $v_{e_h}^{p+1}(2)$  and  $v_{e_g}^{q,r}(1)$  for  $q = p + 1$ , whenever  $e_h \neq e_g$ . Finally, the dashed edge shows that the edges, represented by the edge-vertices incident to it in  $G_1$ , are adjacent in  $H_1$ . In  $G_1$ , length  $m^3 + \log t$  paths exist between  $s_1$  and both of  $s_1^1$  and  $s_1^2$  and between  $t_1$  and both of  $t_1^1$  and  $t_1^2$ . No vertex in this figure contains a token (colored vertices display the colors of the edges in the instance  $(H_1, \phi_1, r_1)$ ).



is connected by paths of length  $(m^3 + \log t) > b$ . Thus, we know that the tokens on the vertices  $v^d$  for  $d \in \{1, \dots, \log t\}$  will have to leave some paths that go from  $a_1$  to  $b_1$  at least through one pair of vertices  $s_\tau$  and  $t_\tau$  for some  $\tau \in [t]$  and can use at most  $\log t$  slides. We know that in  $G \setminus C_\ell$ , no path exists between  $s_\tau$  and  $t_\tau$ . Since no token can reach  $s_\tau$  and  $t_\tau$  in the allocated budget, the remaining slides can only disconnect  $s_\tau$  from  $t_\tau$ . Note also that  $u^{(i,j)} \in C_\ell$ , for  $i \in [2(\kappa - 1)]$  and  $j \in [m - 1]$  as otherwise, a path from  $a_1$  to  $b_1$  that goes through  $s_\tau$ ,  $u^{(i,j)}$  and  $t_\tau$  will remain tokens-free. This implies that at most  $m - 1$  tokens can be slid into any one level  $\{E_\tau^1(\cdot), \dots, E_\tau^{\kappa-1}(\cdot)\}$ . We show via an inductive argument that the set of edges in  $H_\tau$  represented by the vertices in  $\{E_\tau^1(\cdot), \dots, E_\tau^{\kappa-1}(\cdot)\}$  but not in  $C_\ell$  must form a matching  $\mathcal{M}_\tau$  in  $H_\tau$  of size  $\kappa_\tau = \kappa$ , such that for  $e_h, e_g \in \mathcal{M}_\tau$ ,  $\phi_\tau(e_h) \neq \phi_\tau(e_g)$  and the claim follows. Let  $P(q)$  be the proposition that the set  $\mathcal{E}_q$  of edges represented by vertices in  $\{E_\tau^1(\cdot), \dots, E_\tau^q(\cdot)\}$  but not in  $C_\ell$  form a matching such that for  $e_h, e_g \in \mathcal{E}_q$ ,  $\phi_\tau(e_h) \neq \phi_\tau(e_g)$  and that vertices that remain free in  $E_\tau^{q+1}(1)$  for  $q < \kappa - 1$  represent the same edges as the vertices that remain free in  $E_\tau^q(2)$ . We show that  $P(q)$  holds by induction on the levels  $q = \{1, \dots, \kappa - 1\}$ .

We prove the base case by contradiction and assume that a vertex  $v_{e_g}^{1,\tau}(2)$  that remains free in  $E_\tau^1(2)$  either represents an edge  $e_g$  that is incident to an edge  $e_h$  represented by a vertex  $v_{e_h}^{1,\tau}(1)$  that remains free in  $E_\tau^1(1)$  or it holds that  $\phi_\tau(e_g) = \phi_\tau(e_h)$ . This implies that there exists a path between  $s_\tau$  and  $t_\tau$  that goes from  $s_\tau$  to  $s_\tau^1$ , to  $v_{e_h}^{1,\tau}(1)$ ,  $v_{e_g}^{1,\tau}(2)$ ,  $t_\tau^1$  and to  $t_\tau$  and thus  $C_\ell$  is not a solution to  $(G, S, b, a_1, b_1)$ . As for the second part of the statement, assume that a vertex  $v_{e_h}^{1,\tau}(2)$  that remains free in  $E_\tau^1(2)$  does not represent the same edge as any of the vertices that remain free in  $E_\tau^2(1)$ , then there exists a path between  $s_\tau$  and  $t_\tau$  that goes through,  $s_\tau^2$ , then any of the latter vertices, followed by  $v_{e_h}^{1,\tau}(2)$  and  $t_\tau^1$  and thus  $C_\ell$  is not a solution to  $(G, S, b, a_1, b_1)$ . Note that the same arguments used in the base case apply for the inductive step.

In other words, given the second part of the statement, we may assume (for contradiction purposes) that a vertex  $v_{e_g}^{i,\tau}(2)$  for  $i \leq q$  (that remains free in  $E_\tau^i(2)$ ) either represents an edge  $e_g$  that is incident to an edge  $e_h$  represented by a vertex  $v_{e_h}^{i',\tau}(1)$  for  $i' \leq i$  (that remains free in  $E_\tau^{i'}(1)$ ) or it holds that  $\phi_\tau(e_g) = \phi_\tau(e_h)$ . By construction, this implies that there exists a path from  $s_\tau$  and  $t_\tau$  that goes from  $s_\tau$  to  $s_\tau^{i'}$ ,  $v_{e_h}^{i',\tau}(1)$ ,  $v_{e_g}^{i,\tau}(2)$ ,  $t_\tau^{i'}$ , and to  $t_\tau$  and thus  $C_\ell$  is not a solution to  $(G, S, b, a_1, b_1)$ . As for the second part of the statement, assume that a vertex  $v_{e_h}^{q,\tau}(2)$  (that remains free in  $E_\tau^q(2)$ ) does not represent the same edge as any of the vertices that remain free in  $E_\tau^{q+1}(1)$ , then there exists a path between  $s_\tau$  and  $t_\tau$  that goes through,  $s_\tau^{q+1}$ , then any of the latter vertices, followed by  $v_{e_h}^{q,\tau}(2)$  and  $t_\tau^q$  and thus  $C_\ell$  is not a solution to  $(G, S, b, a_1, b_1)$ .

Thus,  $P(\kappa - 1)$  holds and the set  $\mathcal{E}_{\kappa-1}$  of edges represented by vertices in  $\{E_\tau^1(\cdot), \dots, E_\tau^{\kappa-1}(\cdot)\}$  but not  $C_\ell$  form a matching of size  $\kappa$  such that for  $e_h, e_g \in \mathcal{E}_{\kappa-1}$ ,  $\phi_\tau(e_h) \neq \phi_\tau(e_g)$ .  $\triangleleft$

This concludes the proof of the theorem.  $\blacktriangleleft$

---

## References

- 1 Hans L. Bodlaender, Gunther Cornelissen, and Marieke van der Wegen. Problems hard for treewidth but easy for stable gonality. *Computing Research Repository (CoRR)*, abs/2202.06838, 2022. [arXiv:2202.06838](https://arxiv.org/abs/2202.06838).
- 2 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences (J. Comput. Syst. Sci.)*, 75(8):423–434, 2009. doi:10.1016/J.JCSS.2009.04.001.

- 3 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics (SIAM J. Discret. Math.)*, 28(1):277–305, 2014. doi:10.1137/120880240.
- 4 Nicolas Bousquet, Amer E. Mouawad, Naomi Nishimura, and Sebastian Siebertz. A survey on the parameterized complexity of the independent set and (connected) dominating set reconfiguration problems. *Computing Research Repository (CoRR)*, abs/2204.10526, 2022. doi:10.48550/arXiv.2204.10526.
- 5 Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic (Ann. Pure Appl. Log.)*, 84(1):119–138, 1997. doi:10.1016/S0168-0072(95)00020-8.
- 6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 7 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- 8 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.
- 9 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
- 10 Paul Erdős and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society (J. Lond. Math.)*, s1-35(1):85–90, 1960. doi:10.1112/jlms/s1-35.1.85.
- 11 Michael R. Fellows, Mario Grobler, Nicole Megow, Amer E. Mouawad, Vijayaragunathan Ramamoorthi, Frances A. Rosamond, Daniel Schmand, and Sebastian Siebertz. On solution discovery via reconfiguration. In Kobi Gal, Ann Nowé, Grzegorz J. Nalepa, Roy Fairstein, and Roxana Radulescu, editors, *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 700–707. IOS Press, 2023. doi:10.3233/FAIA230334.
- 12 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct pcps for NP. *Journal of Computer and System Sciences (J. Comput. Syst. Sci.)*, 77(1):91–106, 2011. doi:10.1016/J.JCSS.2010.06.007.
- 13 Mario Grobler, Stephanie Maaz, Nicole Megow, Amer E. Mouawad, Vijayaragunathan Ramamoorthi, Daniel Schmand, and Sebastian Siebertz. Solution discovery via reconfiguration for problems in P. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPICs*, pages 76:1–76:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ICALP.2024.76.
- 14 Stephan Kreutzer, Roman Rabinovich, and Sebastian Siebertz. Polynomial kernels and wideness properties of nowhere dense graph classes. *ACM Transactions on Algorithms (ACM Trans. Algorithms)*, 15(2):24:1–24:19, 2019. doi:10.1145/3274652.
- 15 Van Bang Le and Florian Pfender. Complexity results for rainbow matchings. *Theoretical Computer Science (Theor. Comput. Sci.)*, 524:27–33, 2014. doi:10.1016/J.TCS.2013.12.013.
- 16 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/A11040052.
- 17 Michal Pilipczuk, Sebastian Siebertz, and Szymon Torunczyk. On the number of types in sparse graphs. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 799–808. ACM, 2018. doi:10.1145/3209108.3209178.
- 18 Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science (Theor. Comput. Sci.)*, 26:287–300, 1983. doi:10.1016/0304-3975(83)90020-8.