FPT and kernelization algorithms for the Induced Tree problem*

Guilherme C. M. Gomes[1][0000−0002−5164−1460], Vinicius F. dos Santos[1][0000−0002−4608−4559], Murilo V. G. da Silva[2][0000−0002−3392−714X], and Jayme L. Szwarcfiter[3,4][0000−0002−7538−7305]

[1] Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
{gcm.gomes,viniciussantos}@dcc.ufmg.br
[2] Departamento de Informática, Universidade Federal do Paraná, Curitiba, Brazil
murilo@inf.ufpr.br
[3] Universidade Federal do Rio de Janeiro
[4] Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil
jayme@cos.ufrj.br

Abstract. The THREE-IN-A-TREE problem asks for an induced tree of the input graph containing three mandatory vertices. In 2006, Chudnovsky and Seymour [Combinatorica, 2010] presented the first polynomial time algorithm for this problem, which has become a critical subroutine in many algorithms for detecting induced subgraphs, such as beetles, pyramids, thetas, and even and odd-holes. In 2007, Derhy and Picouleau [Discrete Applied Mathematics, 2009] considered the natural generalization to k mandatory vertices with k being part of the input, and showed that it is NP-complete; they named this problem INDUCED TREE, and asked what is the complexity of FOUR-IN-A-TREE. Motivated by this question and the relevance of the original problem, we study the parameterized complexity of INDUCED TREE. We begin by showing that the problem is W[1]-hard when jointly parameterized by the size of the solution and minimum clique cover and, under the Exponential Time Hypothesis, does not admit an $n^{o(k)}$ time algorithm. Afterwards, we use Courcelle's Theorem to prove tractability under cliquewidth, which prompts our investigation into which parameterizations admit single exponential algorithms; we show that such algorithms exist for the unrelated parameterizations treewidth, distance to cluster, and distance to co-cluster. In terms of kernelization, we present a linear kernel under feedback edge set, and show that no polynomial kernel exists under vertex cover nor distance to clique unless NP $\subseteq$ coNP/poly. Along with other remarks and previous work, our tractability and kernelization results cover many of the most commonly employed parameters in the graph parameter hierarchy.

Keywords: induced tree; parameterized complexity; FPT algorithm; polynomial kernel; cross-composition.

* Work partially supported by CAPES, CNPq, and FAPEMIG. Full version permanently available at https://arxiv.org/abs/2007.04468

1 Introduction

Given a graph $G = (V, E)$ and a subset $K \subseteq V(G)$ of size three – here called the set of terminal vertices – the THREE-IN-A-TREE problem consists of finding an induced subtree of G that connects K. Despite the novelty of this problem, it has become an important tool in many detection algorithms, where it usually accounts for a significant part of the work performed during their executions. It was first studied by Chudnovsky and Seymour [13] in the context of theta and pyramid detection, the latter of which is a crucial part of perfect graph recognition algorithms [11] and the former was an open question of interest [10]. Across more than twenty pages, Chudnovsky and Seymour characterized all pairs (G, K) that do not admit a solution, which resulted in a $\mathcal{O}(mn^2)$ time algorithm for the problem on n-vertex, m-edge graphs. Since then, THREE-IN-A-TREE has shown itself as a powerful tool, becoming a crucial subroutine for the fastest known even-hole [9], beetle [9], and odd-hole [12] detection algorithms; to the best of our knowledge, these algorithms often rely on reductions to multiple instances of THREE-IN-A-TREE, e.g. the theta detection algorithm has to solve $\mathcal{O}(mn^2)$ THREE-IN-A-TREE instances to produce its output [34]. Despite its versatility, THREE-IN-A-TREE is not a silver bullet, and some authors discuss quite extensively why they think THREE-IN-A-TREE cannot be used in some cases [14, 39]. Nevertheless, Lai et al. [34] very recently made a significant breakthrough and managed to reduce the complexity of Chudnovsky and Seymour's algorithm for THREE-IN-A-TREE to $\mathcal{O}(m \log^2 n)$, effectively speeding up many major detection algorithms, among other improvements to the number of THREE-IN-A-TREE instances required to solve some other detection problems.

As pondered by Lai et al. [34], the usage of THREE-IN-A-TREE as a go-to solution for detection problems may, at times, seem quite unnatural. In the aforementioned cases, one could try to tackle the problem by looking for constant sized minors or disjoint paths between terminal pairs and then resort to Kawarabayashi et al.'s [32] quadratic algorithm to finalize the detection procedure. The problem is that neither the minors nor the disjoint paths are guaranteed to be induced; to make the situation truly dire, this constraint makes even the most basic problems NP-hard. For instance, Bienstock [1, 2] proved that TWO-IN-A-HOLE and THREE-IN-A-PATH are NP-complete. As such, it is quite surprising that THREE-IN-A-TREE can be solved in polynomial time and be of widespread importance. It is worth to note that the induced subgraph constraint is also troublesome from the parameterized point of view. MAXIMUM MATCHING, for instance, can be solved in polynomial time [23], but if we impose that the matching must be an induced subgraph, the problem becomes W[1]-hard when parameterized by the minimum number of edges in the matching [36].

Derhy and Picouleau [18] were the first to ponder how far we may push for polynomial time algorithms when considering larger numbers of terminal vertices, i.e. they were interested in the complexity of k-IN-A-TREE for $k \geq 4$. They were also the firsts to investigate the more general problem where k is not fixed, which they dubbed INDUCED TREE, for which we have the following formal definition:

Induced Tree
Instance: A graph G and a set $K \subseteq V(G)$.
Question: Is there an induced subtree of G that contains all vertices of K?

Derhy and Picouleau proved in [18] that Induced Tree is NP-complete even on planar bipartite cubic graphs of girth four, but solvable in polynomial time if the girth of the graph is larger than the number of terminals. A few years later, Derhy et al. [19] showed that four-in-a-tree is solvable in polynomial time on triangle-free graphs, while Liu and Trotignon [35] proved that so is k-in-a-tree on graphs of girth at least k; their combined results imply that k-in-a-tree on graphs of girth at least k is solvable in polynomial time; it is important to remark that the running time of the algorithm of Liu and Trotignon [35] has no $n^{f(k)}$ term. In terms of the Induced Path problem, Derhy and Picouleau [18] argued that their hardness reduction also applies to this problem and showed that three-in-a-path is NP-complete even on graphs of maximum degree three. Fiala et al. [24] proved that k-in-a-path, k-Induced Disjoint Paths, and k-in-a-cycle can be solved in polynomial time on claw-free graphs for every fixed k, but that Induced Path, Induced Disjoint Paths, and Induced Cycle are NP-complete in fact, their positive results can be seen as XP algorithms for the latter three problems when parameterized by the number of terminals on claw-free graphs. Another related problem to Induced Tree is the well known Steiner Tree problem, where we want to find a subtree of the input with cost at most w connecting all terminals. Being one of Karp's 21 NP-hard problems [31], Steiner Tree has received a lot of attention over the decades. Relevant to our discussion, however, is its parameterized complexity. When parameterized by the number of terminals, it admits a single exponential time algorithm [22]; the same was proven to be true when treewidth [37] is the parameter [3]. On the other hand, when parameterized by cliquewidth [16], it is paraNP-hard since it is NP-hard even on cliques: we may reduce from Steiner Tree itself and by replacing each non-edge with an edge of cost $w + 1$. As we see below, the parameterized complexity of Steiner Tree and Induced Tree greatly differ.

Our results. Given the hardness results for Induced Tree even on restricted graph classes, we focus our study on the parameterized complexity of the problem. We begin by presenting some algorithmic results for Induced Tree in Section 3, showing that the latter is W[1]-hard when simultaneously parameterized by the number of vertices in the solution ℓ and size of a minimum clique cover q and, moreover, does not admit an $n^{o(\ell+q)}$ time algorithm unless the Exponential Time Hypothesis [30] (ETH) fails. On the positive side, we prove tractability under cliquewidth using Courcelle's Theorem [15], which prompts us, in Section 4, to turn our attention to which parameters allow us to devise single exponential time algorithms for Induced Tree. Using Bodlaender et al.'s dynamic programming optimization machinery [3], we show that such algorithms exist under treewidth, distance to cluster, and distance to co-cluster, all of which are widely used in the literature [7, 17, 21, 28, 33, 41] and were the smallest parameters for which we managed to obtain such algorithms. We con-

clude the study of vertex cover-related parameters in Section 5, where we prove that the problem does not admit a polynomial kernel when simultaneously parameterized by the size of the solution, diameter, and distance to any non-trivial graph class, including the class of independent sets; we also show no such kernel exists when parameterizing by bandwidth. All our negative kernelization results are obtained assuming NP $\not\subseteq$ coNP/poly. In the realm of structural parameters, a natural next step would be to consider the max leaf number parameter. In Section 6 we do so by: (i) showing that max leaf number and feedback edge set are equivalent parameterizations for Induced Tree, and (ii) presenting a kernel with $16q$ vertices and $17q$ edges when we parameterize by the size q of a minimum feedback edge set; aside from our contribution we only know of one other problem for which kernelization under feedback edge set was considered, namely the Edge Disjoint Paths problem [27]. In terms of tractability and kernelization, our results encompass most of the commonly employed parameters of Sorge and Weller's graph parameter hierarchy [38]; we present a summary of our results in Figure 1. To see why the distance to solution parameter sits between vertex cover and feedback vertex set, we refer to the end of Section 3. Due to space restrictions, most proofs have been moved to the appendix.

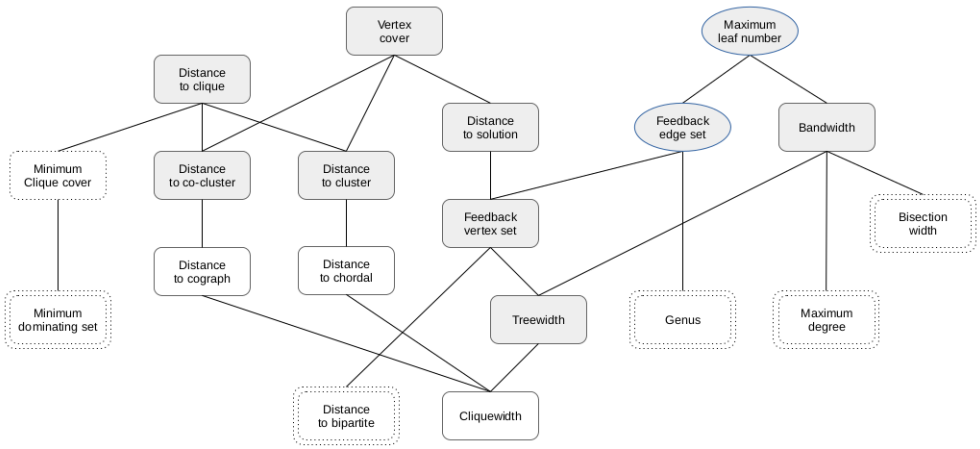

Fig. 1. Hasse diagram of graph parameters and associated results for Induced Tree. Parameters surrounded by shaded ellipses have both single exponential time algorithms and polynomial kernels. Solid boxes represent parameters under which the problem is FPT but does not admit polynomial kernels; if the box is shaded, we have a single exponential time algorithm for that parameterization. A single dashed box corresponds to a W[1]-hard parameterization; double dashed boxes surround parameters under which the problem is paraNP-hard. Aside from the paraNP-hardness for genus, max degree, and distance to bipartite, all results are original contributions proposed in this work.

2 Preliminaries

We refer the reader to [17] for basic background on parameterized complexity, and recall here only some basic definitions. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance $I = (x, q) \in \Sigma^* \times \mathbb{N}$, q is called the *parameter*. A parameterized problem is *fixed-parameter tractable* (FPT) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance (x, q), $\mathcal{A}$ correctly decides whether $I \in L$ in time bounded by $f(q) \cdot |I|^c$; in this case, $\mathcal{A}$ is called an FPT *algorithm*. A fundamental concept in parameterized complexity is that of *kernelization*; see [25] for a recent book on the topic. A kernelization algorithm, or just *kernel*, for a parameterized problem Π takes an instance (x, q) of the problem and, in time polynomial in $|x| + q$, outputs an instance (x', q') such that $|x'|, q' \leqslant g(q)$ for some function g, and $(x, q) \in \Pi$ if and only if $(x', q') \in \Pi$. Function g is called the *size* of the kernel and may be viewed as a measure of the "compressibility" of a problem using polynomial-time pre-processing rules. A kernel is called *polynomial* (resp. *quadratic, linear*) if $g(q)$ is a polynomial (resp. quadratic, linear) function in q. A breakthrough result of Bodlaender et al. [4] gave the first framework for proving that some parameterized problems do not admit polynomial kernels, by establishing so-called *composition algorithms*. Together with a result of Fortnow and Santhanam [26], this allows to exclude polynomial kernels under the assumption that NP $\not\subseteq$ coNP/poly, otherwise implying a collapse of the polynomial hierarchy to its third level [40].

All graphs in this work are finite and simple. We use standard graph theory notation and nomenclature for our parameters; for any undefined terminology in graph theory we refer to [6]. We denote the degree of vertex v on graph G by $\deg_G(v)$, and the set of natural numbers $\{1, 2, \ldots, t\}$ by $[t]$. A graph is a *cluster graph* if each of its connected components is a clique, while a *co-cluster graph* is the complement of a cluster graph. The *distance to cluster* (*co-cluster*) of a graph G, is the size of the smallest set $U \subseteq V(G)$ such that $G \setminus U$ is a cluster (co-cluster) graph. As defined in [8], a set $U \subseteq V(G)$ is an $\mathcal{F}$-*modulator* of G if $G \setminus U$ belongs to the graph class $\mathcal{F}$. When the context is clear, we omit the qualifier $\mathcal{F}$. For cluster and co-cluster graphs, one can decide if G admits a modulator of size q in time FPT on q [7].

3 Fixed-parameter tractability and intractability

While it has been known for some time that Induced Tree is NP-complete even on planar bipartite cubic graphs, it is not known to be even in XP when parameterized by the natural parameter: the number of terminals. We take a first step with a negative result about this parameterization, ruling out the existence of an FPT algorithm unless FPT = W[1]; in fact, we show this for stronger parameterization: the maximum size of the induced tree that should contain the set of k terminal vertices K and the size of a minimum clique cover. The proof of the following two theorems can be found in Appendix A.

Theorem 1. Induced Tree *is* W[1]-hard *when simultaneously parameterized by the number of vertices of the induced tree and size of a minimum clique cover. Moreover, unless ETH fails, there is no* $n^{o(k)}$ *time algorithm for* Induced Tree.

Since the natural parameters offer little to no hope of fixed-parameter tractability, to obtain parameterized algorithms we turn our attention to the broad class of structural parameters. Our first positive result is a direct application of textbook MSO_1 formulae. We remark that the edge-weighted version of two-in-a-tree is strongly NP-complete [18].

Theorem 2. *When parameterized by cliquewidth,* Induced Tree *can be solved in* FPT *time. Moreover, the same holds true even for the edge-weighted version of* Induced Tree *where we want to minimize the weight of the tree.*

Towards showing that the minimum number of vertices we must delete to obtain a solution sits between feedback vertex set and vertex cover in Figure 1, let $S \subset V(G)$ be such that $G \setminus S$ is a solution. First, note that S is a feedback vertex set of G; for the other inequality, take a vertex cover C of G and note that placing two vertices of $G \setminus C$ with the same neighborhood in C either generates a cycle in the solution or only one of them suffices – even if we have many terminals, we need to keep only two of them – so $|S| \leq |C| + 2^{|C|+1}$. In terms of paraNP-hardness results, we can easily show that Induced Tree is paraNP-hard when parameterized by bisection width[5]: to reduce from the problem to itself, we pick any terminal of the input and append to it a path with as many vertices as the original graph to obtain a graph with bisection width one. Similarly, when parameterizing by the size of a minimum dominating set and again reducing from Induced Tree to itself, we add a new terminal adjacent to any vertex of K and a universal vertex, which can never be part of the solution since it forms a triangle with the new terminal and its neighbor.

4 Single exponential time algorithms

All results in this section rely on the *rank based approach* of Bodlaender et al. [3]. Proofs and other technical details of this section are deferred to Appendix B. Even though our problem is unweighted, we found it convenient to solve the slightly more general weighted problem below when parameterizing Induced Tree by treewidth [37].

Light Connecting Induced Subgraph
Instance: A graph G, a set of k terminals $K \subseteq V(G)$, and two integers ℓ, f.
Question: Is there a connected induced subgraph of G on $\ell + k$ vertices and at most f edges that contains K?

[5] The width of a bipartition (A, B) of $V(G)$ is the number of edges between the parts. The bisection width is the minimum width of all bipartitions of $V(G)$ such that $|A| \leq |B| \leq |A| + 1$.

Note that an instance (G, K) of Induced Tree is positive if and only if there is some integer ℓ where the Light Connecting Induced Subgraph instance $(G, K, \ell, \ell + k - 1)$ is positive. Our goal is to use the number of edges in the solution to Light Connecting Induced Subgraph as the cost of a partial solution in a dynamic programming algorithm. This shall be particularly useful for join nodes during our treewidth algorithm, as we may resort to the optimality of the solution to guarantee that the resulting induced subgraph of a join operation is acyclic. As usual, we assume that we are given a nice tree decomposition [17] in the input to our algorithm.

Theorem 3. *There is an algorithm for* Light Connecting Induced Subgraph *that, given a nice tree decomposition of width t of the n-vertex input graph G, runs in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$.*

Corollary 1. *There is an algorithm for* Induced Tree *that, given a nice tree decomposition of width t of the n-vertex input graph G, runs in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$.*

We also use the framework of Bodlaender et al. [3] for the next two results. For Theorem 4, we observe that a clique may have at most two vertices in any solution to Induced Tree. For Theorem 5, observe that at most two parts of a complete multipartite subgraph may intersect the solution and, if this is the case, one part has at most one vertex, otherwise we would have an induced C_4.

Theorem 4. *There is an algorithm for* Induced Tree *that runs in time $2^{\mathcal{O}(q)}n^{\mathcal{O}(1)}$ on graphs with distance to cluster at most q graphs.*

Theorem 5. *There is an algorithm for* Induced Tree *that runs in time $2^{\mathcal{O}(q)}n^{\mathcal{O}(1)}$ where q is the distance to co-cluster.*

5 Kernelization lower bounds

In this section, we apply the cross-composition framework of Bodlaender et al. [5] to show that, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, Induced Tree does not admit a polynomial kernel under bandwidth, nor when parameterized by the distance to any graph class with at least one member with t vertices for each integer t, which we collectively call non-trivial classes. We say that an NP-hard problem R *OR-cross-composes* into a parameterized problem L if, given t instances $\{y_1, \ldots, y_t\}$ of R, we can construct, in time polynomial in $\sum_{i\in[t]} |y_i|$, an instance (x, k) of L that satisfies $k \leq p(\max_{i\in[t]} |y_i| + \log t)$ for some polynomial $p(\cdot)$ and admits a solution if and only if at least one instance y_i of R admits a solution; we say that R *AND-cross-composes* into L if the first two conditions hold but (x, k) has a solution if and only if all t instances of R admit a solution. We prove Theorem 6 in Appendix C using an AND-cross-composition from Induced Tree to itself, while Theorem 7 and Corollary 2 are proved in Appendix D by an OR-cross-composition from Hamiltonian Path.

Theorem 6. *When parameterized by bandwidth,* Induced Tree *does not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

Theorem 7. Induced Tree *does not admit a polynomial kernel when parameterized by the number of vertices of the induced tree, and size of a minimum vertex cover unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Corollary 2. *For every non-trivial graph class* $\mathcal{G}$, Induced Tree *does not admit a polynomial kernel when parameterized by the number of vertices of the induced tree and size of a minimum* $\mathcal{G}$*-modulator unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

6 A linear kernel for Feedback Edge Set

In this section, we prove that Induced Tree admits a linear kernel when parameterized by the size q of a minimum feedback edge set. Throughout this section, we denote our input graph by G, the set of terminals by K, and the tree obtained by removing the edges of a minimum size feedback edge set F by $T(F)$. Note that, if G is connected and F is of minimum size, $G \setminus F$ is a tree; we may safely assume the first, otherwise we either have that (G, K) is a negative instance if K is spread across multiple connected components of G, or there must be some edge of F that merges two connected components of $G \setminus F$ and does not create a cycle, contradicting the minimality of F. The algorithm we describe works in two steps: it first finds a feedback edge set F that minimizes the number of edges incident to vertices of degree two in $T(F)$, then compresses long induced paths of G. We denote the set of leaves of a tree H by $\mathsf{leaves}(H)$. Missing proofs are given in Appendix E.

Reduction Rule 1. *If G has a vertex v of degree one, remove v and, if $v \in K$, add the unique neighbor of v in G to K.*

Proof of safeness of Rule 1. Safeness follows from the fact that a degree one vertex is in the solution if and only if its unique neighbor also is. □

Observation 1. *After exhaustively applying Rule 1, for every minimum feedback edge set F of G, $T(F)$ has at most $2q$ leaves. Moreover, $T(F)$ has at most as many vertices of degree at least three as leaves.*

We begin with any minimum feedback edge set F of G. We partition $T(F) \setminus \mathsf{leaves}(T(F))$ into (D_2, D_*) according to the degree of the vertices of G in $T(F)$: $v \in D_2$ if and only if $\deg_{T(F)}(v) = 2$. For $u, v, f \in V(G)$, we say that u *F-links* v to f if $v = u$ or if $T(F) \setminus \{u\}$ has no $v - f$ path. We say that vertices u, f are an *F-pair* if the set of internal vertices of the unique $u - f$ path $P_F(u, f)$ of $T(F)$ is entirely contained in D_2; we denote the set of internal vertices by $P_F^*(u, f)$.

Reduction Rule 2. *Let $u, f, w_1, w_2 \in V(G)$ be such that u, f form an F-pair, $w_1 \neq w_2 \neq u \neq w_1$, w_2 is the unique neighbor of f that F-links it to u and w_1 F-links w_2 to u. If $fw_1 \in F$, remove edge fw_1 from F and add edge fw_2 to F.*

Proof of safeness of Rule 2. Let $F' = F \setminus \{fw_1\}$ and note that w_2 F-links f to w_1; as such, edge fw_2 is in the unique cycle of $G \setminus F'$, so $F'' = F' \cup \{fw_2\}$ is

a feedback edge set of G of size q. Furthermore, w_2 is the only vertex that has fewer neighbors in $T(F'')$ than in $T(F)$; since w_2 had two neighbors in $T(F)$, and F'' is a minimum feedback edge set of G, w_2 is a leaf of $T(F'')$, so it holds that $\mathsf{leaves}\,(T(F)) \subset \mathsf{leaves}\,(T(F''))$. □

Reduction Rule 2 guarantees that there are no edges in F between vertices of the paths between F-pairs, otherwise $|\mathsf{leaves}\,(T(F))|$ would not be maximal.

Reduction Rule 3. *Let $f, u, v \in V(G)$ be such that $v \notin \mathsf{leaves}\,(T(F)) \cup P_F(u, f)$, u, f form an F-pair, and $|P_F(u, f)| \geq 4$. If there are adjacent vertices $w_1, w_2 \in P_F^*(u, f)$ with $vw_1 \in F$ and w_2 F-linking v and w_1, remove edge vw_1 from F and add edge w_1w_2 to F.*

Proof of safeness of Rule 3. Let $F' = F \setminus \{w_1w_2\}$. Since $G \setminus F'$ has one more edge than $T(F)$ and w_2 F-links v and w (see Figure 2), the unique cycle of $G \setminus F'$ contains edge w_1w_2, so $F'' = F' \cup \{vw_1\}$ is a feedback edge set of G of size q. Since neither v nor w are leaves of $T(F)$ and $w_2 \in D_2$, $\deg_{T(F'')}(w_2) = 1$, so it holds that $|\mathsf{leaves}\,(T(F))| < |\mathsf{leaves}\,(T(F''))|$. □

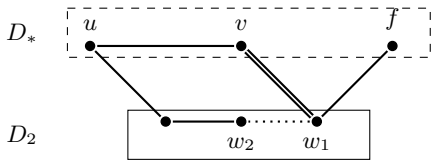

Fig. 2. Example for Reduction Rule 3, where the thick edge vw_1 is removed from F and the dotted edge w_1w_2 added to F.

Note that, in Reduction Rule 3, the only properties that we exploit are that $v, w_1, w_2 \notin \mathsf{leaves}\,(T(F))$ and that there is at least one pair of vertices between v and f in D_2. So even if $v = u$, $u \in D_2$, or $f \in \mathsf{leaves}\,(T(F))$, we can apply Rule 3. Essentially, if Rule 3 is not applicable, for each edge $e \in F$ that contains a vertex w_1 of D_2 as an endpoint, either e has a leaf of $T(F)$ as its other endpoint, or w_1 is adjacent to two vertices not in D_2.

Reduction Rule 4. *Let $f, u, v, z, w_2 \in V(G)$ be such that $v \in \mathsf{leaves}\,((T(F)) \setminus \{f\}$, $w_2 \in D_2$ is the unique neighbor of v in $T(F)$, u, f and z, v are F-pairs, and u F-links f to z. If there is some $w_1 \in P_F^*(u, f)$ with $vw_1 \in F$, remove vw_2 from F and add vw_1.*

Proof of safeness of Rule 4. Let $F' = F \setminus \{vw_1\}$. Since $T(F)$ is a tree and w_2 F-links w_1 and v, edge vw_2 is contained in the unique cycle of $G \setminus F'$; consequently, $F'' = F' \cup \{vw_2\}$ is a feedback edge set of G of size q, but it holds that the degrees of v and w_2 in $T(F'')$ are equal to one. Since $w_2 \notin \mathsf{leaves}\,(T(F))$, we have that $\mathsf{leaves}\,(T(F)) \subset \mathsf{leaves}\,(T(F''))$. □

Our analysis for Rule 4 works even if $u = z$ or $z = w_2$: what is truly crucial is that $w_2 \in D_2$ and that $v \neq f$. We present an example of the general case in Figure 3.

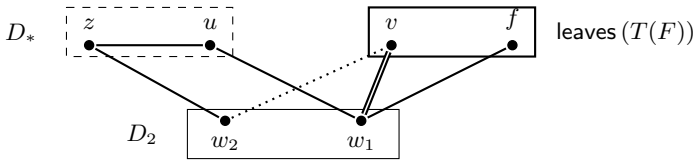

Fig. 3. Example for Reduction Rule 4, where the thick edge vw_1 is removed from F and the dotted edge vw_2 is added to F.

Reduction Rule 5. *Let $f, u, v, z, w_2, w_3 \in V(G)$ be such that $v \in \mathsf{leaves}\,((T(F))$, $w_3 \in N_{T(F)}(u) \cap P_F(u, f)$, $w_2w_3, vz \in E(T(F))$, u, f is an F-pair, $z \in D_*$, and u F-links f to v. If v is adjacent to some $w_1 \in P_F(u, f) \setminus \{w_2\}$ that F-links w_2 to f, remove vw_1 from F and add w_2w_3 to F.*

Proof of safeness of Rule 5. Let $F' = F \setminus \{vw_3\}$. Since w_1 F-links w_2 to f, we have that w_2 F-links w_1 to v, so edge w_2w_3 belongs to the unique cycle of $G \setminus F'$ and, consequently, $F'' = F' \cup \{w_2w_3\}$ is a feedback edge set of G of size q. Since $\deg_{T(F)}(w_3) = \deg_{T(F)}(w_2) = 2$, $\deg_{T(F'')}(w_3) = \deg_{T(F'')}(w_2) = 1$, however, v is adjacent to both z and w_1 in $T(F'')$, so it holds that $\mathsf{leaves}\,(T(F'')) = \mathsf{leaves}\,(T(F)) \cup \{w_2, w_3\} \setminus \{v\}$. □

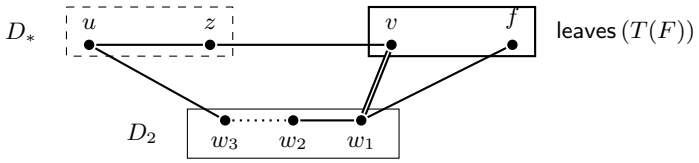

Fig. 4. Example for Reduction Rule 5, where the dotted edge w_2w_3 is added to F and the thick edge vw_1 is removed from F.

We present an example of Reduction Rule 5 in Figure 4. Our next lemma guarantees that the exhaustive application of rules 2 through 5 finds a set of paths in $T(F)$ that have many vertices of degree two in G; essentially, at this point, we are done minimizing the number of incident edges to vertices of D_2.

Lemma 1. *Let $a, b \in V(G)$ be an F-pair such that $a, b \notin D_2$, $|P_F^*(a, b)| \geq 5$, and let w be one of its inner vertices at distance at least three from both a and b. If none of the rules between Rule 2 and Rule 5 are applicable, then* $\deg_G(w) = \deg_{T(F)}(w)$.

At this point, paths between F-pairs are mostly the same as in G: only the two vertices closest to each endpoint may be adjacent to some leaves of $T(F)$, while all others have degree two in G. We say that u, f are a *strict* F-pair if for every $w \in P_F^*(u, f)$, $\deg_G(w) = 2$.

Reduction Rule 6. *Let $u, f \in V(G)$ be a strict F-pair. If there are adjacent vertices $w_1, w_2 \in P_F^*(u, f)$ such that either $w_1, w_2 \in K$ or $w_1, w_2 \notin K$, add a new vertex w^* to G that is adjacent to $N_G(w_1) \cup N_G(w_2) \setminus \{w_1, w_2\}$ and remove both w_1, w_2 from G. If $w_1, w_2 \in K$, set w^* as a terminal vertex.*

Proof of safeness of Rule 6. Correctness follows directly from the hypotheses that $w_1 \in K$ if and only if $w_2 \in K$ and that both are degree two vertices. So, in a minimal solution H to (G, K), either both vertices are in H or neither is in H. For the converse, any minimal solution H' to the reduced instance (G', K') either has w^*, in which H is obtained by replacing w^* with both w_1 and w_2, or $w^* \notin V(H)$, in which case H' is a solution to (G, K). □

Reduction Rule 7. *Let $u, f \in V(G)$ be a strict F-pair such that $P_F^*(u, f) \geq 4$. If Rule 6 is not applicable, replace $P_F^*(u, f)$ with three vertices a, t, b so that a is adjacent to u, b to f, and t to both a and b. Furthermore, t is a terminal of the new graph if and only if $K \cap P_F^*(u, f) \neq \emptyset$.*

Proof of safeness of Rule 7. Let G' and K' be, respectively, the graph and set of terminals obtained after the application of the rule. Suppose H is a minimal solution to the Induced Tree instance (G, K), i.e every vertex of H is contained in a path between two terminals. Note that, if $P_F^*(u, f) \cap V(H) = \emptyset$, H is also a solution to the instance (G', K'); as such, for the remainder of this paragraph, we may assume w.l.o.g. that $P_F^*(u, f) \cap V(H) \neq \emptyset$ and that $u \in V(H)$. If $P_F(u, f) \setminus \{u\} \nsubseteq V(H)$, $H' = H \cup \{a, t\} \setminus P_F^*(u, f)$ is a solution to (G', K'): since at least one vertex of $P_F(u, f)$ is not in $V(H)$ and every $w \in P_F^*(u, f)$ has degree two in G, the subpaths of $P_F(u, f)$ in H are used solely for the collection of terminal vertices of $P_F(u, f)$; consequently, H' is an induced tree of G' that contains all elements of K'. On the other hand, if $P_F(u, f) \subseteq V(H)$, $H' = H \cup \{a, t, b\} \setminus P_F^*(u, f)$ is a solution to (G', K'); to see that this is the case, note that $H \setminus P_F^*(u, f)$ is a forest with exactly two trees where u and f are in different connected components since $P_F(u, f)$ is the unique path between them in H, and $K' \subseteq V(H')$ since $P_F^* \subseteq V(H)$ and $K \setminus P_F^*(u, f) \subseteq V(H) \setminus P_F^*(u, f)$.

For the converse, let H' be a minimal solution to (G', K'). If $\{a, t, b\} \subseteq V(H')$, $H = H' \cup \{P_F^*(u, f)\} \setminus \{a, t, b\}$ is a solution of (G, K), as we are replacing one path consisting solely of degree two vertices with another that satisfies the same property. If $a \in V(H')$ but $b \notin V(H')$, then $t \in K'$ (recall that H' is minimal) and $u \in V(H)$, implying that there is at least one terminal vertex in $P_F^*(u, f)$. We branch our analysis in the following subcases, where $v \in P_F^*(u, f) \cap N_G(f)$:

- If $v \notin K$, then $H = H' \cup P_F^*(u, f) \setminus \{v\} \setminus \{a, t\}$ is a solution to (G, K): all terminals of $P_F^*(u, f)$ are contained in $P_F^*(u, v)$ and no cycle is generated since all vertices of the path $P_F^*(u, v)$ have degree two.

- If $v \in K$ but $f \notin V(H')$, $H = H' \cup P_F^*(u,f) \setminus \{a,t\}$ is a solution to (G,K): we cannot create any new cycle since $f \notin V(H)$ and u, f form a strict F-pair, moreover all terminals of $P_F^*(u,f)$ are contained in H.
- If $v \in K$ and $f \in V(H')$, there is at least one non-terminal vertex $w \in P_F^*(u,v)$ since Rule 6 is not applicable to $P_F(u,f)$. As such, we set $H = H' \cup P_F^*(u,w) \cup P_F^*(w,f) \setminus \{a,t\}$ and obtain a solution to (G,K).

Finally, if $\{a,t,b\} \cap V(H') = \emptyset$, it follows immediately from the assumption that H' is a solution to (G',K') that H' is also a solution to (G,K). □

We are now ready to state our kernelization theorem.

Theorem 8. *When parameterized by the size q of a feedback edge set,* Induced Tree *admits a kernel with $16q$ vertices and $17q$ edges that can be computed in $\mathcal{O}(q^2 + qn)$ time.*

7 Concluding Remarks

In this work, we performed an extensive study of the parameterized complexity of Induced Tree and the existence of polynomial kernels for the problem, motivated by the relevance of three-in-a-tree in subgraph detection algorithms and related work on Induced Tree itself, specially the question on the complexity of four-in-a-tree posed by Derhy and Picouleau [18]. We presented multiple positive and negative results for Induced Tree, of which we highlight its W[1]-hardness under the natural parameter, the linear kernel under feedback edge set, and the nonexistence of a polynomial kernel under vertex cover/distance to clique. The main question about the complexity of k-in-a-tree for *fixed* k, however, remains open; our hardness result showed that there is no $n^{o(k)}$ time algorithm under the ETH, but no XP algorithm is known to exist. There are also no known running time lower bounds for the parameters we study, and determining whether or not we can obtain $2^{o(q)}n^{\mathcal{O}(1)}$ time algorithms seems a feasible research direction; still in terms of algorithmic results, it would be quite interesting to see how we can avoid Courcelle's Theorem to get an algorithm when parameterizing by cliquewidth. The natural investigation of k-in-a-tree on different graph classes may provide some insights on how to tackle particular cases, such as four-in-a-tree; this study has already been started in [20] and in others – such as cographs – may even be trivial, but many other cases may be quite challenging and much still remains to be done.

References

1. Dan Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85 – 92, 1991. URL: http://www.sciencedirect.com/science/article/pii/0012365X9190098M, doi:https://doi.org/10.1016/0012-365X(91)90098-M.
2. Dan Bienstock. Corrigendum: To: D. bienstock, "On the complexity of testing for odd holes and induced odd paths" Discrete Mathematics 90 (1991) 85–92. *Discrete Mathematics*, 102(1):109, 1992. URL: http://www.sciencedirect.com/science/article/pii/0012365X9290357L, doi:https://doi.org/10.1016/0012-365X(92)90357-L.
3. Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86 – 111, 2015. 40th International Colloquium on Automata, Languages and Programming (ICALP 2013). URL: http://www.sciencedirect.com/science/article/pii/S0890540114001606, doi:https://doi.org/10.1016/j.ic.2014.12.008.
4. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423 – 434, 2009. URL: http://www.sciencedirect.com/science/article/pii/S0022000009000282, doi:https://doi.org/10.1016/j.jcss.2009.04.001.
5. Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *Proc. of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *LIPIcs*, pages 165–176, 2011.
6. J.A. Bondy and U.S.R Murty. *Graph Theory*. Springer Publishing Company, Incorporated, 1st edition, 2008.
7. Anudhyan Boral, Marek Cygan, Tomasz Kociumaka, and Marcin Pilipczuk. A fast branching algorithm for cluster vertex deletion. *Theory of Computing Systems*, 58(2):357–376, 2016. doi:10.1007/s00224-015-9631-7.
8. Leizhen Cai. Parameterized complexity of vertex colouring. *Discrete Applied Mathematics*, 127(3):415 – 429, 2003. URL: http://www.sciencedirect.com/science/article/pii/S0166218X02002421, doi:https://doi.org/10.1016/S0166-218X(02)00242-1.
9. Hsien-Chih Chang and Hsueh-I Lu. A faster algorithm to recognize even-hole-free graphs. *Journal of Combinatorial Theory, Series B*, 113:141 – 161, 2015. URL: http://www.sciencedirect.com/science/article/pii/S0095895615000155, doi:https://doi.org/10.1016/j.jctb.2015.02.001.
10. Maria Chudnovsky and Rohan Kapadia. Detecting a theta or a prism. *SIAM Journal on Discrete Mathematics*, 22(3):1164–1186, 2008. arXiv:https://doi.org/10.1137/060672613, doi:10.1137/060672613.
11. Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164(1):51–229, 2006. URL: http://www.jstor.org/stable/20159988.
12. Maria Chudnovsky, Alex Scott, Paul Seymour, and Sophie Spirkl. Detecting an odd hole. *J. ACM*, 67(1), January 2020. doi:10.1145/3375720.
13. Maria Chudnovsky and Paul Seymour. The three-in-a-tree problem. *Combinatorica*, 30(4):387–417, 2010. doi:10.1007/s00493-010-2334-4.

14. Maria Chudnovsky, Paul Seymour, and Nicolas Trotignon. Detecting an induced net subdivision. *Journal of Combinatorial Theory, Series B*, 103(5):630 – 641, 2013. URL: http://www.sciencedirect.com/science/article/pii/S0095895613000531, doi:https://doi.org/10.1016/j.jctb.2013.07.005.
15. Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012. doi:10.1017/CBO9780511977619.
16. Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77 – 114, 2000. URL: http://www.sciencedirect.com/science/article/pii/S0166218X99001845, doi:https://doi.org/10.1016/S0166-218X(99)00184-5.
17. Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 3. Springer, 2015.
18. Nicolas Derhy and Christophe Picouleau. Finding induced trees. *Discrete Applied Mathematics*, 157(17):3552 – 3557, 2009. Sixth International Conference on Graphs and Optimization 2007. URL: http://www.sciencedirect.com/science/article/pii/S0166218X09000663, doi:https://doi.org/10.1016/j.dam.2009.02.009.
19. Nicolas Derhy, Christophe Picouleau, and Nicolas Trotignon. The four-in-a-tree problem in triangle-free graphs. *Graphs and Combinatorics*, 25(4):489, 2009. doi:10.1007/s00373-009-0867-3.
20. Vinícius F. dos Santos, Murilo V.G. da Silva, and Jayme L Szwarcfiter. The k-in-a-tree problem for chordal graphs. *Matemática Contemporânea*, 44:1–10, 2015.
21. Martin Doucha and Jan Kratochvíl. Cluster vertex deletion: A parameterization between vertex cover and clique-width. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012*, pages 348–359, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
22. S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230010302, doi:10.1002/net.3230010302.
23. Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.
24. Jiří Fiala, Marcin Kamiński, Bernard Lidický, and Daniël Paulusma. The k-in-a-path problem for claw-free graphs. *Algorithmica*, 62(1):499–519, 2012. doi:10.1007/s00453-010-9468-z.
25. Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.
26. Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct pcps for np. *Journal of Computer and System Sciences*, 77(1):91 – 106, 2011. Celebrating Karp's Kyoto Prize. URL: http://www.sciencedirect.com/science/article/pii/S0022000010000917, doi:https://doi.org/10.1016/j.jcss.2010.06.007.
27. Robert Ganian and Sebastian Ordyniak. The power of cut-based parameters for computing edge disjoint paths. In Ignasi Sau and Dimitrios M. Thilikos, editors, *Graph-Theoretic Concepts in Computer Science*, pages 190–204, Cham, 2019. Springer International Publishing.
28. Guilherme C. M. Gomes, Matheus R. Guedes, and Vinícius F. dos Santos. Structural parameterizations for equitable coloring, 2019. arXiv:1911.03297.

29. Sang il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514 – 528, 2006. URL: http://www.sciencedirect.com/science/article/pii/S0095895605001528, doi:https://doi.org/10.1016/j.jctb.2005.10.006.
30. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001. URL: http://www.sciencedirect.com/science/article/pii/S0022000000917276, doi:https://doi.org/10.1006/jcss.2000.1727.
31. Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2_9.
32. Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424 – 435, 2012. URL: http://www.sciencedirect.com/science/article/pii/S0095895611000712, doi:https://doi.org/10.1016/j.jctb.2011.07.004.
33. Christian Komusiewicz, Dieter Kratsch, and Van Bang Le. Matching cut: Kernelization, single-exponential time fpt, and exact exponential algorithms. volume 283, pages 44 – 58, 2020. URL: http://www.sciencedirect.com/science/article/pii/S0166218X19305530, doi:https://doi.org/10.1016/j.dam.2019.12.010.
34. Kai-Yuan Lai, Hsueh-I Lu, and Mikkel Thorup. Three-in-a-tree in near linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 1279–1292, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384235.
35. W. Liu and N. Trotignon. The k-in-a-tree problem for graphs of girth at least k. *Discrete Applied Mathematics*, 158(15):1644 – 1649, 2010. URL: http://www.sciencedirect.com/science/article/pii/S0166218X10002131, doi:https://doi.org/10.1016/j.dam.2010.06.005.
36. Hannes Moser and Somnath Sikdar. The parameterized complexity of the induced matching problem. *Discrete Applied Mathematics*, 157(4):715 – 727, 2009. URL: http://www.sciencedirect.com/science/article/pii/S0166218X08003211, doi:https://doi.org/10.1016/j.dam.2008.07.011.
37. Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986. URL: http://www.sciencedirect.com/science/article/pii/0196677486900234, doi:https://doi.org/10.1016/0196-6774(86)90023-4.
38. Manuel Sorge and Mathias Weller. The graph parameter hierarchy. *unpublished manuscript*, 2019.
39. Nicolas Trotignon and Kristina Vušković. A structure theorem for graphs with no cycle with a unique chord and its consequences. *Journal of Graph Theory*, 63(1):31–67, 2010. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.20405, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/jgt.20405, doi:10.1002/jgt.20405.
40. Chee K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287 – 300, 1983. URL: http://www.sciencedirect.com/science/article/pii/0304397583900208, doi:https://doi.org/10.1016/0304-3975(83)90020-8.
41. Édouard Bonnet and Florian Sikora. The graph motif problem parameterized by the structure of the input graph. *Discrete Applied Mathematics*, 231:78 – 94, 2017. Algorithmic Graph Theory on the Adriatic Coast. URL: http://www.sciencedirect.com/science/article/pii/S0166218X1630539X, doi:https://doi.org/10.1016/j.dam.2016.11.016.

Appendix

A Fixed-parameter tractability and intractability

While it has been known for some time that Induced Tree is NP-complete even on planar bipartite cubic graphs, it is not known to be even in XP when parameterized by the natural parameter: the number of terminals. We take a first step with a negative result about this parameterization, ruling out the existence of an FPT algorithm unless FPT = W[1]; in fact, we show this for stronger parameterization: the maximum size of the induced tree that should contain the set of k terminal vertices K and the size of a minimum clique cover.

Theorem (1). Induced Tree *is* W[1]-hard *when simultaneously parameterized by the number of vertices of the induced tree and size of a minimum clique cover. Moreover, unless ETH fails, there is no* $n^{o(k)}$ *time algorithm for* Induced Tree.

Proof of Theorem 1. Our reduction is from Multicolored Independent Set parameterized by the number of color classes ℓ. Formally, let H be the input to our source problem such that $V(H)$ is partitioned into ℓ color classes $\{V_1, \ldots, V_\ell\}$ and each V_i induces a clique on H. Our instance of Induced Tree (G, K) is such that $K = \{v_0, \ldots, v_\ell\}$, $V(G) = K \cup V(H)$, every edge of H is also in G, each $v_i \in K$ has $N_G(v_i) = V_i$, and $N(v_0) = V(H)$, so $k = \ell + 1$.

If I is a solution to Multicolored Independent Set, $I \cup K$ is a solution to (G, K): there are no cycles since I is also an independent set of G, v_0 connects all vertices of I, and each other terminal is connected to exactly one vertex of I. For the converse, if T is a solution to (G, K), we claim that $I = T \setminus K$ is an independent set of size ℓ. To see that this is the case, note that: (i) $T \cap V(H)$ must be independent, otherwise they would form a triangle with v_0; and (ii) $|T \cap V_i| = 1$ because each V_i is a clique and the only way to connect v_i to v_0 is by picking at least one vertex of V_i. Since $|K| = \ell + 1$ we have that the solution size is at most $2\ell + 1$ and, since $Q_i = V_i \cup \{v_i\}$ is a clique, we can cover G with the k cliques $\{\{v_0\}\} \cup \bigcup_{i \in [k]} \{Q_i\}$. The second statement follows from the fact that Multicolored Independent Set has no $n^{o(k)}$ time algorithm under ETH [17] and that our reduction exhibits a linear relation between the parameters. □

Since the natural parameters offer little to no hope of fixed-parameter tractability, to obtain parameterized algorithms we turn our attention to the broad class of structural parameters. Our first positive result is a direct application of textbook MSO_1 formulae.

Theorem (2). *When parameterized by cliquewidth,* Induced Tree *can be solved in* FPT *time. Moreover, the same holds true even for the edge-weighted version of* Induced Tree *where we want to minimize the weight of the tree.*

Proof of Theorem 2. Let (G, K) be the input to Induced Tree, $e(u, v)$ the relation that is true if and only if $uv \in E(G)$, R be the binary relation represented by the formula $\varphi_R(u, v, Y) = u \in Y \wedge v \in Y \wedge (e(u, v) \vee e(v, u))$, and $TC[R; x, y]$ be

the reflexive and transitive closure of R. If a q-expression is not given in the input, then we obtain a q'-expression, where $q' \leq 2^{3q+2}-1$, using the algorithm of Oum and Seymour [29]. As such, $\mathsf{conn}(Y) = \forall x, y(x \in Y \wedge y \in Y \Rightarrow TC[R; x, y])$ is true if and only if $G[Y]$ is connected [15]. Similarly, formula $\mathsf{cycle}(X) = \exists x, y, z \in X(x \neq y \wedge y \neq z \wedge x \neq z \wedge e(x, z) \wedge e(y, z) \wedge \exists Y \subset X(z \notin Y \wedge x \in Y \wedge y \in Y \wedge \mathsf{conn}(Y))$ is true if and only if $G[X]$ has a cycle [15]. Putting the previous two formulae together, formula $\mathsf{indtree}(K) = \exists S(K \subseteq S \wedge \mathsf{conn}(S) \wedge \neg\mathsf{cycle}(S))$ is true if and only if there is some superset of K that is connected and acyclic. By Courcelle's Theorem [15], if G has n vertices, cliquewidth at most q, and we are given a q-expression of G, then one can determine in $f(q)n^{\mathcal{O}(1)}$ time if G satisfies $\mathsf{indtree}(K)$ for some set K of terminals.

For the second statement, let $f : E(G) \mapsto \mathbb{Z}$ be the weights of the edge-weighted version of Induced Tree, and $w : E(K_n) \mapsto \mathbb{Z}$ be a function from the edges of the complete graph on n vertices to the integers where $w(uv) = f(uv)$ whenever $uv \in E(G)$, and $w(uv) = 0$ otherwise. The LinMSO_1 formula $\mathsf{minindtree}(K) = \min_{S \subseteq V(G)} \sum_{u,v \in S} w(uv) : \mathsf{indtree}(K)$ concludes the proof. □

Towards showing that the minimum number of vertices we must delete to obtain a solution sits between feedback vertex set and vertex cover in Figure 1, let $S \subset V(G)$ be such that $G \setminus S$ is a solution. First, note that S is a feedback vertex set of G; for the other inequality, take a vertex cover C of G and note that placing two vertices of $G \setminus C$ with the same neighborhood in C either generates a cycle in the solution or only one of them suffices – even if we have many terminals, we need to keep only two of them – so $|S| \leq |C| + 2^{|C|+1}$. In terms of paraNP-hardness results, we can easily show that Induced Tree is paraNP-hard when parameterized by bisection width[6]: to reduce from the problem to itself, we pick any terminal of the input and append to it a path with as many vertices as the original graph to obtain a graph with bisection width one. Similarly, when parameterizing by the size of a minimum dominating set and again reducing from Induced Tree to itself, we add a new terminal adjacent to any vertex of K and a universal vertex, which can never be part of the solution since it forms a triangle with the new terminal and its neighbor.

B Single exponential time algorithms

Let U be a finite set, $\Pi(U)$ denote the set of all partitions of U, and $\sqsubseteq$ be the coarsening relation defined on $\Pi(U)$, i.e given two partitions $p, q \in \Pi(U)$, $p \sqsubseteq q$ if and only if each block of q is contained in some block of p. It is known that $\Pi(U)$ together with $\sqsubseteq$ form a lattice, upon which we can define the usual *join* operator $\sqcup$ and *meet* operator $\sqcap$ [3]. The join operation $p \sqcup q$ outputs the unique partition z where two elements are in the same block of z if and only if they are in the same block of p or q. The result of the meet operation $p \sqcap q$ is the

[6] The width of a bipartition (A, B) of $V(G)$ is the number of edges between the parts. The bisection width is the minimum width of all bipartitions of $V(G)$ such that $|A| \leq |B| \leq |A| + 1$.

unique partition such that each block is formed by the non-empty intersection between a block of p and a block of q. Given a subset $X \subseteq U$ and $p \in \Pi(U)$, $p_{\downarrow X} \in \Pi(X)$ is the partition obtained by removing all elements of $U \setminus X$ from p, while, for $Y \supseteq U$, $p_{\uparrow Y} \in \Pi(Y)$ is the partition obtained by adding to p a singleton block for each element in $Y \setminus U$. For $X \subseteq U$, we shorthand by $U[X]$ the partition where one block is precisely $\{X\}$ and all other are the singletons of $U \setminus X$; if $X = \{a, b\}$, we use $U[ab]$.

A set of *weighted partitions* of a ground set U is defined as $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$. To speed up dynamic programming algorithms for connectivity problems, the idea is to only store a subset $\mathcal{A}' \subseteq \mathcal{A}$ that preserves the existence of at least one optimal solution. Formally, for each possible extension $q \in \Pi(U)$ of the current partitions of $\mathcal{A}$ to a valid solution, the optimum of $\mathcal{A}$ relative to q is denoted by $\mathsf{opt}(q, \mathcal{A}) = \min\{w \mid (p, w) \in \mathcal{A}, p \sqcup q = \{U\}\}$. $\mathcal{A}'$ *represents* $\mathcal{A}$ if $\mathsf{opt}(q, \mathcal{A}') = \mathsf{opt}(q, \mathcal{A})$ for all $q \in \Pi(U)$. The key result of Bodlaender et al. [3] is given by Theorem 9.

Theorem 9 (3.7 of [3]). *There exists an algorithm that, given $\mathcal{A}$ and U, computes $\mathcal{A}'$ in time $|\mathcal{A}|2^{(\omega-1)|U|}|U|^{\mathcal{O}(1)}$ and $|\mathcal{A}'| \leq 2^{|U|-1}$, where ω is the matrix multiplication constant.*

A function $f : 2^{\Pi(U)\times\mathbb{N}} \times Z \mapsto 2^{\Pi(U)\times\mathbb{N}}$ is said to *preserve representation* if $f(\mathcal{A}', z) = f(\mathcal{A}, z)$ for every $\mathcal{A}, \mathcal{A}' \in \Pi(U) \times \mathbb{N}$ and $z \in Z$; thus, if one can describe a dynamic programming algorithm that uses only transition functions that preserve representation, it is possible to obtain $\mathcal{A}'$. In the following lemma, let $\mathsf{rmc}(\mathcal{A}) = \{(p, w) \in \mathcal{A} \mid \nexists (p, w') \in \mathcal{A}, w' < w\}$.

Lemma 2 (Proposition 3.3 and Lemma 3.6 of [3]). *Let U be a finite set and $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$. The following functions preserve representation and can be computed in $|\mathcal{A}| \cdot |\mathcal{B}| \cdot |U|^{\mathcal{O}(1)}$ time.*

Union. *For $\mathcal{B} \in \Pi(U) \times \mathbb{N}$, $\mathcal{A} \uplus \mathcal{B} = \mathsf{rmc}(\mathcal{A} \cup \mathcal{B})$.*

Insert. *For $X \cap U = \emptyset$, $\mathsf{ins}(X, \mathcal{A}) = \{(p_{\uparrow X \cup U}, w) \mid (p, w) \in \mathcal{A}\}$.*

Shift. *For any integer w', $\mathsf{shift}(w', \mathcal{A}) = \{(p, w + w') \mid (p, w) \in \mathcal{A}\}$.*

Glue. *Let $\hat{U} = U \cup X$, then $\mathsf{glue}(X, \mathcal{A}) = \mathsf{rmc}\left(\left\{(\hat{U}[X] \sqcup p_{\uparrow \hat{U}}, w) \mid (p, w) \in \mathcal{A}\right\}\right)$.*

Project. *$\mathsf{proj}(X, \mathcal{A}) = \mathsf{rmc}\left(\{(p_{\downarrow \overline{X}}, w) \mid (p, w) \in \mathcal{A}, \forall u \in \overline{X} : \exists v \in \overline{X} : p \sqsubseteq U[uv]\}\right)$, if $X \subseteq U$.*

Join. *If $\hat{U} = U \cup U'$, $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$ and $\mathcal{B} \in \Pi(U') \times \mathbb{N}$, then $\mathsf{join}(\mathcal{A}, \mathcal{B}) = \mathsf{rmc}\left(\{(p_{\uparrow \hat{U}} \sqcup q_{\uparrow \hat{U}}, w + w') \mid (p, w) \in \mathcal{A}, (q, w') \in \mathcal{B}\}\right)$.*

B.1 Treewidth

A *tree decomposition* of a graph G is a pair $\mathbb{T} = (T, \mathcal{B} = \{B_j \mid j \in V(T)\})$, where T is a tree and $\mathcal{B} \subseteq 2^{V(G)}$ is a family where: $\bigcup_{B_j \in \mathcal{B}} B_j = V(G)$; for every edge $uv \in E(G)$ there is some B_j such that $\{u, v\} \subseteq B_j$; for every $i, j, q \in V(T)$, if q is in the path between i and j in T, then $B_i \cap B_j \subseteq B_q$. Each $B_j \in \mathcal{B}$ is called

a *bag* of the tree decomposition. G has treewidth has most t if it admits a tree decomposition such that no bag has more than t vertices. For further properties of treewidth, we refer to [37]. After rooting T, G_x denotes the subgraph of G induced by the vertices contained in any bag that belongs to the subtree of T rooted at bag x. An algorithmically useful property of tree decompositions is the existence of a *nice tree decomposition* that does not increase the treewidth of G.

Definition 1 (Nice tree decomposition). *A tree decomposition $\mathbb{T}$ of G is said to be* nice *if its tree is rooted at, say, the empty bag $r(T)$ and each of its bags is from one of the following four types:*

1. Leaf node*: a leaf x of $\mathbb{T}$ with $B_x = \emptyset$.*
2. Introduce vertex node*: an inner bag x of $\mathbb{T}$ with one child y such that $B_x \setminus B_y = \{u\}$.*
3. Forget node*: an inner bag x of $\mathbb{T}$ with one child y such that $B_y \setminus B_x = \{u\}$.*
4. Join node*: an inner bag x of $\mathbb{T}$ with two children y, z such that $B_x = B_y = B_z$.*

Theorem (3). *There is an algorithm for* Light Connecting Induced Subgraph *that, given a nice tree decomposition of width t of the n-vertex input graph G rooted at the forget node for some terminal $r \in K$, runs in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$.*

Proof of Theorem 3. Let (G, K, ℓ, f) be an instance of Light Connecting Induced Subgraph. For each bag x, we compute the table $g_x(S, \ell') \subseteq \Pi(S) \times \mathbb{N}$, where $S \subseteq B_x$ contains the vertices of B_x that must be present in a solution and ℓ' is the number of vertices we allow in the induced subgraphs of G_x; each weighted partition $(p, w) \in g_x(S, \ell')$ corresponds to a choice of an induced subgraph of G_x with ℓ' vertices, $w + |E(G[S])|$ edges, and connected components given by the blocks of p. If $|S| > \ell'$, we define $g_x(S, \ell') = \emptyset$. After every operation, we apply the algorithm of Theorem 9.

Leaf node. Since $B_x = \emptyset$, the only possible connecting induced subgraph is precisely the empty graph, so we define:

$$g_x(\emptyset, \ell') = \begin{cases} \{(\emptyset, 0)\}, & \text{if } \ell' = 0; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Introduce node. Let y be the child bag of x and $B_x \setminus B_y = \{v\}$. We compute $g_x(S, \ell')$ as follows, where $\mathcal{A}_y(S, \ell', v) = \mathsf{ins}(\{v\}, g_y(S \setminus \{v\}, \ell' - 1))$:

$$g_x(S, \ell') = \begin{cases} \mathsf{glue}\left(N[v] \cap S, \mathcal{A}_y(S, \ell', v)\right), & \text{if } v \in S; \\ g_y(S, \ell'), & \text{if } v \notin S \cup K. \\ \emptyset, & \text{otherwise.} \end{cases}$$

On the third case above, if $v \in K$ but $v \notin S$, we are not including a terminal into the induced subgraph, thus we cannot accept any partition with support S as valid. If $v \notin K \cup S$, then no changes are necessary, since the only vertex of G_x not in G_y is not considered for the solution. Finally, for the first case,

since $v \in S$, we must extend each partition of $g_y(S \setminus \{v\}, \ell' - 1)$ to the ground set S (which we achieve by the insert operation); however, since we are looking for an induced subgraph, we must use every edge between v and the neighbors of v in S, merging the connected components containing them. Unlike in some connectivity problems such as Steiner Tree, we only count the edges *within* S while processing forget nodes; this shall simplify the join operation considerably, as we do not need to worry about repeatedly counting edges inside the current bag.

Forget node. Let y be the child bag of x and v be the forgotten vertex. The transition is directly computed by:

$$g_x(S, \ell') = g_y(S, \ell') \uplus \mathsf{shift}(|N(v) \cap S|, \mathsf{proj}(\{v\}, g_y(S \cup \{v\}, \ell')))$$

If v is not used in a partial solution, $g_y(S, \ell')$ already correctly contains all the partial solutions where v is not used; on the other hand, if v was used in some solution, we must eliminate v from the partitions where it appears; however, we only keep the partitions that do not lose a block, otherwise we would have a connected component (represented by v) that shall never be connected to the remainder of the subgraph and, thus, cannot be extended to a valid solution.

Join node. If y, z are the children of bag x, we compose its table by the equation:

$$g_x(S, \ell') = \biguplus_{\substack{\ell_1 + \ell_2 \,=\, \ell' + |S| \\ |S| \,\leq\, \ell_1, \ell_2}} \mathsf{join}(g_y(S, \ell_1), g_z(S, \ell_2))$$

Where the union operator runs over all integer values satisfying the system $\ell_1 + \ell_2 = \ell' + |S|,\ |S| \leq \ell_1, \ell_2$; since we do not know how many vertices were used on the partial solutions of each subtree, we must try every combination to obtain all the partial solutions for the subtree rooted at x. Since we force the vertices in S to be present in the solutions to the subtree rooted at bag x, combining two partial solutions, represented by $(p, w) \in g_y(S, \ell_1)$ and $(q, w') \in g_y(S, \ell_2)$, corresponds to uniting the set of edges of the respective partial solutions $G(p), G(q)$, which results in a merger of connected components. Since the edges of S have not been counted towards the weights w, w', $G(p \sqcup q)$ has exactly $w + w' + |E(G[S])|$ edges. This is precisely the definition of the join operation.

In order to obtain the answer to the problem, we look at the child x of the forget node for terminal r, that is, the child of the root of the tree, and check if $g_x(\{r\}, \ell + |K|) \neq \emptyset$. In the affirmative, note that there is only one entry $(\{\{r\}\}, w) \in g(\{r\}, \ell + |K|)$ and that the graph that connects all the terminals using $\ell + |K|$ vertices has exactly w edges, since $E(G[\{r\}])0\emptyset$.

For an introduce bag x with child y, the time taken to compute all entries of g_x is of the order of $\ell \sum_{i=0}^{|B_x|} \binom{|B_x|}{i} 2^{\omega i} t^{\mathcal{O}(1)} \leq n(1 + 2^{\omega i}) t t^{\mathcal{O}(1)}$; the term $2^{\omega i}$ comes from the time needed to execute the algorithm of Theorem 9 upon a initial set of size 2^i. For join nodes, the intermediate $\sqcup$ operation may yield a set of size 4^i, so we have that the tables can be computed in time $\ell^3 \sum_{i=0}^{|B_x|} \binom{|B_x|}{i} 2^{(\omega+1)i} t^{\mathcal{O}(1)} \leq (1 + 2^{(\omega+1)})^t n^{\mathcal{O}(1)}$. □

Corollary 3. *There is an algorithm for* Induced Tree *that, given a nice tree decomposition of width t of the n-vertex input graph G rooted at the forget node for some terminal $r \in K$, runs in time $2^{\mathcal{O}(t)}n^{\mathcal{O}(1)}$.*

B.2 Distance to cluster

We now show that parameterizing by the distance to cluster q also yields an FPT algorithm. Throughout this section, G is the input graph, U is the cluster modulator, and $\mathcal{C} = \{C_1, \ldots, C_r\}$ are the maximal cliques of $G \setminus U$. We also use the framework developed by Bodlaender et al. [3] to optimize our dynamic programming algorithm.

Theorem (4). *There is an algorithm for* Induced Tree *that runs in time $2^{\mathcal{O}(q)}n^{\mathcal{O}(1)}$ on graphs with distance to cluster at most q graphs.*

Proof of Theorem 4. Suppose we are given the instance (G, K) and the q-vertex cluster modulator $U \subseteq V(G)$. We begin by guessing a subset $K \cap U \subseteq S \subseteq U$ of vertices that will be present in a solution for the problem. Now, given S, we execute the following pre-processing step: for each clique $C_i \in \mathcal{C}$, we discard all but one vertex of each maximal set of true twins; this way, we limit the size of $C_i^* = C_i \setminus K$ to 2^q.

An entry of our dynamic programming table $f_S(i, c, \ell) \subseteq \Pi(S) \times \mathbb{N}$ is a set of partial solutions of $G_i = G[K \cup S \bigcup_{j=1}^{i} C_j]$, each of which uses exactly c vertices of C_i^* and induces a subgraph of G_i on ℓ vertices. Note that we cannot use more than two vertices of each clique, so we only consider $c \in \{0, 1, 2\}$. In each $(p, w) \in f_S(i, c, \ell)$, each block of p corresponds to the vertices of S that lie in the same connected component of G_i, and w is the number of edges used in the respective induced subgraph. Our transition is given by the following equation, where $W(S, X) = |N(X) \cap (S \cup K)| + |E(G[X])|$; if either $c + |K \cap C_i| > 2$, $c > \ell$, or there is some vertex in $K \cap C_i$ that has no neighbor in S and $c = 0$, we define $f_S(i, c, \ell) = \emptyset$.

$$f_S(i, c, \ell) = \biguplus_{j=0}^{2} \biguplus_{X \in \binom{C_i^*}{c}} \mathsf{glue}_{W(S,X)}(N_S(X), f_S(i-1, j, \ell - c))$$

The above defines f_S for all $(i, c, \ell) \in [r] \times \{0, 1, 2\} \times [n]$; we extend f_S to include the base case $f_S(0, 0, |S \cup K|) = \{(p(S, K), E(G[S \cup K])\}$, where $p(S, K) \in \Pi(S)$ is the partition obtained by gluing together the connected components of $G[S]$ which have a common neighbor in K; for all other entries, $f_S(r+1, c, \ell) = \emptyset$. Our goal now is to show that there is a solution to our problem using the vertices of S if and only if $(\{S\}, w) \in f_S(r, c, \ell)$, for some pair $\ell \geq |S \cup K|$, $c \geq |K \cap C_1|$, such that $w = \ell - 1$. To do so, we first prove that $f(i, c, \ell)$ contains all partitions of S that represent all possible induced subgraphs of G_i on ℓ vertices that use c vertices of C_i^*, and that use as few edges as possible.

By induction, suppose that this holds for every entry $f_S(i-1, a, b)$. If $c + |K \cap C_i| > 2$ or $c > \ell$, either we want to use more than two vertices of C_i, which

certainly implies that there is a copy of K_3 in the solution, or we want to use more than ℓ vertices of C_i^*, which is equally impossible, so $f_S(i, c, \ell) = \emptyset$. If $c = 0$, we have that a weighted partition (p, w) is valid in G_i if and only if it is valid in G_{i-1}, since we use no vertices in $V(G_i) \setminus V(G_{i-1}$; thus, $(p, w) \in f_S(i, 0, \ell)$ if and only if $(p, w) \in f_S(i-1, j, \ell)$ for some $j \in \{0, 1, 2\}$. Otherwise, suppose we want to add a subset of vertices $X \subseteq C_i^*$ to a partial solution H, which is represented by $(p, w) \in f_S(i-1, a, \ell - |X|)$. In this case, since $N(X) \setminus C_i \subseteq U$ and $U \cap V(H) \subseteq S$, we have that X can only reduce the number of connected components of H if $N(X)$ intersects two distinct blocks of p. Thus, the connected components of $G[V(H) \cup X]$ are represented, precisely, by $\mathsf{glue}(N_S(X), p)$ and the only new edges used are those between X and $S \cup K$, and the ones internal to X; this is precisely the shift accounted by $W(S, X)$. The minimality of w for an entry $(p, w) \in f(i, c, \ell)$ is guaranteed by the rmc operation imbued in both glue and $\uplus$. Note that if there is some entry $(p, w) \in f_S(r, c, \ell)$ for some c such that $p = \{S\}$ and $w = \ell - 1$, this means that there is an induced subgraph of graph that has S contained in a connected component, uses ℓ vertices and $\ell - 1$ edges, and thus, must be an induced tree of G. That it connects all vertices of K follows from the fact that $f_S(0, 0, |S \cup K|)$ contains $p(S, K)$ and, in every clique C_i that contains a vertex of K with no neighbor in S, we force that at least one vertex of C_i must be picked in a solution.

In terms of complexity, after every glue or $\uplus$ operation, we apply the algorithm of Theorem 9. For each tuple (S, i, c, ℓ, j), we do so up to $|C_i^*|^2 \leq 2^{2q}$ times per tuple, which implies in a time requirement time of the order of $2^{2q} \cdot 2^{q-1} 2^{(\omega-1)q} q^{\mathcal{O}(1)} \leq 2^{(\omega+3)q} q^{\mathcal{O}(1)}$. Since we have $2^q n^{\mathcal{O}(1)}$ tuples, our algorithm runs in $2^{(\omega+4)q} n^{\mathcal{O}(1)}$ time. □

B.3 Distance to co-cluster

We can use the result on distance to cluster to solve the problem on graphs with distance to co-cluster at most q without much effort, as we see in the following proposition.

Theorem (5). *There is an algorithm for* Induced Tree *that runs in time* $2^{\mathcal{O}(q)} n^{\mathcal{O}(1)}$ *where* q *is the distance to co-cluster.*

Proof of Theorem 5. Suppose we are given a co-cluster modulator U of the input graph G and let $\mathcal{I} = \{I_1, \ldots, I_r\}$ be the family of independent sets of $G - U$. Since $G - U$ is a complete multipartite graph, if we pick vertices of three distinct elements of $\mathcal{I}$, we will form a K_3 in the induced subgraph. Moreover, for each pair $I_i, I_j \in \mathcal{I}$, at most one of them may have more than one vertex in any solution, the induced subgraph would contain a C_4. This implies that K can intersect $V(G - U)$ in at most three vertices and at most two independent sets. If this intersection has size three, for each $K \cap U \subseteq S \subseteq U$, we can easily verify in polynomial time if $G[S \cup K]$ is a tree. Otherwise, for each pair $I_i, I_j \in \mathcal{I}$ such that $K \subseteq U \cup I_i \cup I_j$, we guess which one of them will have more than one vertex in the solution, say I_i, and which vertex $v \in I_i$ will be in the solution, with the

restriction that $K \subseteq U \cup I_j \cup \{v\}$. Now, the graph $G' = G[U \cup I_j \cup \{v\}]$ has a cluster modulator $U \cup \{v\}$, and we can apply the algorithm of Theorem 4 on it to decide if there is an induced tree of G' connecting K. It follows from the observations that there is a valid induced subtree of G if and only if for some choice I_i, I_j and $v \in I_i$ G' has one such induced subgraph. □

C Bandwidth

Theorem (6). *When parameterized by bandwidth,* Induced Tree *does not admit a polynomial kernel unless* NP ⊆ coNP/poly.

Proof of Theorem 6. We are going to show that Induced Tree AND-cross-composes into itself. Let $\mathcal{H} = \{(H_1, K_1), \ldots, (H_t, K_t)\}$ be a set of instances of Induced Tree where each graph has n vertices, $\ell \geq 3$ of which are terminals. The input (G, K) to Induced Tree parameterized by bandwidth is constructed as follows: G is initially the disjoint union of the t input graphs and $K = \bigcup_{i \in [t]} K_i$; now, for each $i \in [t]$, take two distinct terminals $v_1(i), v_2(i)$ and add edge $v_2(i)v_1(i+1)$ for every $i \in [t-1]$. Essentially, we are organizing the H_i's in a path.

Suppose now that every H_i has a solution T_i and note that $T = \bigcup_{i \in [t]} V(T_i)$ is a solution to (G, K): T is a tree and every terminal in K has a path to another. For the converse, take a solution T to (G, K) and let $T_i = T \cap V(H_i)$. To see that T_i is in fact a solution to (H_i, K_i), it suffices to observe that there can be no path between two vertices of H_i that contains vertices that do not belong to H_i. As to the bandwidth, we claim that it is at most $n - 1$: we can set each vertex of H_i in the interval $[n(i-1), ni-1]$ arbitrarily as long as $v_1(i)$ is place at $n(i-1)$ and $v_2(i)$ at $ni - 1$, obtaining the mapping f. Consequently, every edge $ab \in E(H_i)$ satisfies $|f(a) - f(b)| \leq n$ and each edge $v_2(i)v_1(i+1)$ satisfies $f(v_1(i+1)) - f(v_2(i)) = n(i+1-1) - (ni-1) = 1$. □

D Vertex Cover

We show that Hamiltonian Path on cubic graphs OR-cross-composes into Induced Tree parameterized by vertex cover and number of terminals. Our construction, however, can be trivially adapted to different parameterizations, such as distance to clique. In both cases, we heavily rely on the original gadget by Derhy and Picouleau [18], but make some modifications to suit our needs. Let H be an instance of Hamiltonian Path on cubic graphs. The *Derhy-Picouleau graph* of H, which we denote by $\mathsf{DP}(H)$, is constructed as follows: for each $v_i \in V(H)$, add to $\mathsf{DP}(H)$ one copy T_i of the gadget depicted in Figure 5 and, for each edge $v_iv_j \in E(H)$, connect one of the black vertices of T_i to one of the black vertices of T_j so that the degree of each black vertex of $\mathsf{DP}(H)$ is three. We say that T_i and T_j are *adjacent* if there is an edge between a black vertex α_i of T_i and a black vertex β_j of T_j, where $\{\alpha, \beta\} \subset \{a, b, c\}$. The set of mandatory vertices of $\mathsf{DP}(H)$ is the set of gray vertices.

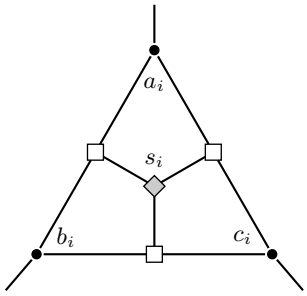

Fig. 5. Vertex gadget T_i for vertex v_i. Vertex s_i is the only terminal of this gadget; white vertices are part of an independent set of maximum size.

Before presenting the composition itself, we need to make some slight modifications to $\mathsf{DP}(H)$, to obtain what we dubbed the *representative graph* of H. Ultimately, our goal is to overlay the multiple instances of Hamiltonian Path and, through an instance selector gadget, force the appropriate graph to emerge from the confounding structure.

Representative Graph Our key modification to $\mathsf{DP}(H)$ is to replace the edge between black vertices with edge gadgets. Suppose that $v_iv_j \in E(H)$, $i < j$, and that $\alpha_i\beta_j \in \mathsf{DP}(H)$. We replace the latter edge with the four vertex gadget $e(i, j, \alpha, \beta)$ as in Figure 6. Note that $e(i, j, \alpha, \beta)$ and $e(i, j, \beta, \alpha)$ are different gadgets whenever $\alpha \neq \beta$. By doing this for every edge of H, we obtain the *representative graph* of H, denoted by $\mathsf{Rep}(H)$. Intuitively, if c_i, b_j are in the solution of the Induced Tree instance given by $\mathsf{DP}(H)$, then g_{ij}^{cb} is not in the solution of the instance whose input is $\mathsf{Rep}(H)$. If either c_i or b_j are not in the solution, g_{ij}^{cb} acts as a garbage collector and is used to connect s_j and s_{ij}^{cb}.

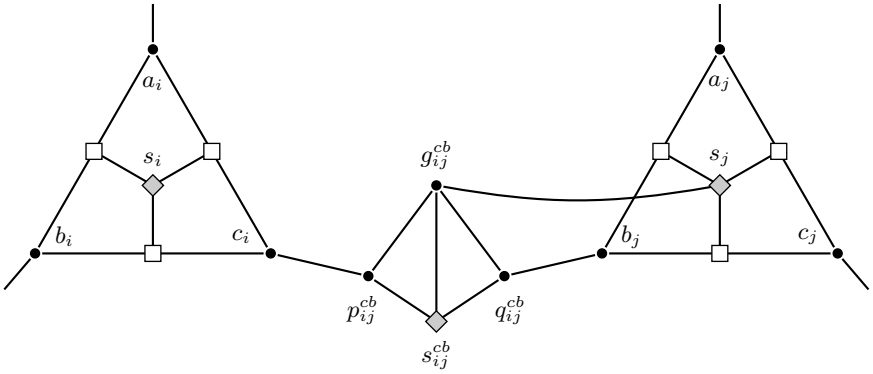

Fig. 6. Edge gadget $e(i, j, c, b)$ for edge $v_iv_j \in E(H)$ $(i < j)$.

Lemma 3. *There is an induced tree connecting the terminal vertices of* $\mathsf{DP}(H)$ *if and only if there is an induced tree connecting the terminal vertices of* $\mathsf{Rep}(H)$.

Proof. Let S be a solution to $\mathsf{DP}(H)$; we construct the solution S' to $\mathsf{Rep}(H)$ as follows. S' contains every vertex in S. For every pair of adjacent black vertices α_i, β_j add $s_{ij}^{\alpha\beta}$ to S'; if both α_i and β_j are in S, add $\{p_{ij}^{\alpha\beta}, q_{ij}^{\alpha\beta}\}$ to S'; otherwise add $g_{ij}^{\alpha\beta}$ to S'; this concludes the definition of S'. To see that S' induces a tree of $\mathsf{Rep}(H)$, note that each path $\langle s_i$, white vertex, α_i, β_j, white vertex, $s_j\rangle$ effectively had edge $\alpha_i\beta_j$ replaced by an induced P_5; furthermore, $g_{ij}^{\alpha\beta}$ is in S' if and only if $\{\alpha_i, \beta_j\} \not\subseteq S$, so no cycle can be formed in the edge gadget; since S induces a tree of $\mathsf{DP}(H)$, we conclude that S' induces a tree of $\mathsf{Rep}(H)$. Finally, S' contains all terminal (gray) vertices of $\mathsf{Rep}(H)$: all such vertices also in $\mathsf{DP}(H)$ were already connected, while the new ones are either included in the induced P_5's with endpoints α_i, β_j, or are connected by $g_{ij}^{\alpha\beta}$ to s_j (assuming $i < j$).

For the converse, suppose $S' \subset V(\mathsf{Rep}(H))$ induces a tree of $\mathsf{Rep}(H)$. Note that we can assume that S' is minimal; in particular, we may safely assume that every black α_i vertex in S' is used to connect s_i to some other s_j or, at the very least, to some terminal of an edge gadget. With this restriction in mind, we obtain our solution S to $\mathsf{DP}(H)$ by setting $S := S' \cap V(\mathsf{DP}(H))$; note that, if S' is not minimal, there could be a pair of vertices $\alpha_i \in V(T_i)$, $\beta_j \in v(T_j)$ with $\alpha_i\beta_j \in E(\mathsf{DP}(H))$, which would imply that S was not acyclic. Towards showing that S induces a tree of $\mathsf{DP}(H)$, let s_i, s_j be two terminals of $\mathsf{DP}(H)$ and $P_{\mathsf{Rep}}(i, j)$ be the unique path between them in the subgraph of $\mathsf{Rep}(H)$ induced by S'. We claim that there is no vertex $g_{\ell r}^{\alpha\beta}$ in $P_{\mathsf{Rep}}(i, j)$: $g_{\ell r}^{\alpha\beta} \in S'$ implies that it is the unique neighbor of $s_{\ell r}^{\alpha\beta}$ in S', otherwise S' would not induce a tree of $\mathsf{Rep}(H)$. Consequently, $P_{\mathsf{Rep}}(i, j) \cap V(\mathsf{DP}(H))$ induces a path of $\mathsf{DP}(H)$ and, since $\bigcup_{i,j\in[n]} P_{\mathsf{Rep}}(i, j)$ induce a tree of $\mathsf{Rep}(H)$, we conclude that $\bigcup_{i,j\in[n]} P_{\mathsf{Rep}}(i, j) \cap V(\mathsf{DP}(H))$ induces a tree of $\mathsf{DP}(H)$. □

The OR-cross-composition For the remainder of this section, we assume that we are given t instances $\mathcal{H} = \{H_1, \ldots, H_t\}$ of Hamiltonian Path such that each $H_\ell \in \mathcal{H}$ is cubic and has n vertices, and the input we construct to Induced Tree is the graph G. What we are going to do now is overlay the representative graphs of $\mathcal{H}$ while avoiding additional vertex gadgets and maintaining only a few additional copies of the edge gadgets. To this end, if $\alpha_i\beta_j \in E(\mathsf{DP}(H_\ell))$, we say that edge v_iv_j of H_ℓ is *represented* by the ordered pair (α, β); in a slight abuse of notation, we write $\mathsf{Rep}_\ell(v_iv_j) = (\alpha, \beta)$, where $\{\alpha, \beta\} \subset \{a, b, c\}$. Formally, G initially has n copies $\{T_1, \ldots, T_n\}$ of the vertex gadget given in Figure 5. For each $\ell \in [t]$, define $\mathcal{E}_\ell$ as the set of edge gadgets of $\mathsf{Rep}(H_\ell)$, i.e $e(i, j, \alpha, \beta) \in \mathcal{E}_\ell$ if and only if $v_iv_j \in E(H)$ and $\mathsf{Rep}_\ell(v_iv_j) = (\alpha, \beta)$. We update G to include all vertex gadgets and all edge gadgets contained in some $\mathcal{E}_\ell$. It is critical to note that $\bigcup_{\ell\in[t]} \mathcal{E}_\ell$ has $\mathcal{O}(n^2)$ elements. For our instance selector gadget, we have a copy of $K_{2,t}$ with bipartition (X, Y), where each of the t vertices of Y corresponds to one instance in $\mathcal{H}$, both vertices of X are terminal vertices, and one of them is identified with the terminal vertex s_1 of T_1. To conclude the construction of G, for each $y_\ell \in Y$ and edge gadget $e(i, j, \alpha, \beta) \notin \mathcal{E}_\ell$, we add all edges between y_ℓ and the four vertices of $e(i, j, \alpha, \beta)$, as we show in Figure 7.

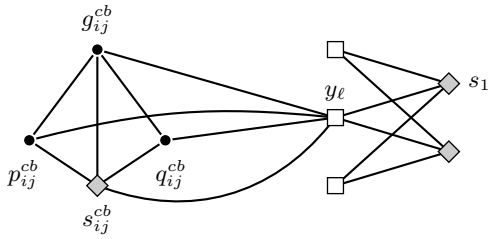

Fig. 7. Interaction between the instance selector gadget and the edge gadget $e(i, j, c, b)$ $(i < j)$, if $e(i, j, c, b) \notin \mathcal{E}_\ell$.

Lemma 4. *The graph G has a vertex cover of size $\mathcal{O}(n^2)$ and $\mathcal{O}(n^2)$ terminals.*

Proof. Note that G has $7n$ vertices in vertex gadgets, at most $18(n^2-3n)$ vertices in edge gadgets, and $t + 1$ vertices in the instance selector gadget (recall that one vertex of X is identified with a terminal of T_1). Since Y is an independent set, $V(G) \setminus Y$ is a vertex cover with $\mathcal{O}(n^2)$ elements. For the last part of the statement, it suffices to observe that the set of terminals of G is a subset of $V(G) \setminus Y$. □

Lemma 5. *There is no induced tree of G connecting all terminal vertices with zero or more than one vertex of Y. Moreover, if $y_\ell \in Y$ is fixed, the graph obtained after removing X, Y, and all vertices that are in a triangle with y_ℓ is precisely $\mathsf{Rep}(H_\ell)$.*

Proof. If no vertex of Y is picked, then there is no path between the two terminals $s_1, x \in X$ precisely because $N(x) = Y$. If two vertices $y_\ell, y_p \in Y$ are picked, then $\{x, y_\ell, s_1, y_p\}$ is an induced C_4.

Now, let $e(i, j, \alpha, \beta)$ be an edge gadget of G. For the second part of the statement, recall that $V(e(i, j, \alpha, \beta)) \subseteq N(y_\ell) \setminus X$ if and only if $e(i, j, \alpha, \beta) \notin \mathcal{E}_\ell$, i.e the vertices W_ℓ of G that form a triangle with y_ℓ are precisely those that belong to the edge gadgets $e(i, j, \alpha, \beta)$ *not* present in $\mathsf{Rep}(H_\ell)$. As such, the induced subgraph of G that remains after the removal of W_ℓ, X and Y is composed precisely of the vertex gadgets and $\mathcal{E}_\ell$; since no extra edges were added within either group of gadgets or between them, we have that $G \setminus (W_\ell \cup X \cup Y) = \mathsf{Rep}(H_\ell)$. □

Theorem (7). Induced Tree *does not admit a polynomial kernel when parameterized by the number of vertices of the induced tree, and size of a minimum vertex cover unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Theorem 7 is a direct consequence of Lemmas 3, 4, and 5. As for Corollary 2, we observe that there is nothing special about set Y of our instance selector gadget being an independent set; the key feature is that only one of its vertices may be used in a solution; as such, we may freely encode a member of whichever non-trivial graph class we are interested in $G[Y]$; a graph class is non-trivial if it contains a graph of t vertices, for every $t \in \mathbb{N}$.

Corollary (2). *For every non-trivial graph class* $\mathcal{G}$, Induced Tree *does not admit a polynomial kernel when parameterized by the number of vertices of the induced tree and size of a minimum* $\mathcal{G}$*-modulator unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

E Feedback edge set

Lemma (1). *Let $a, b \in V(G)$ be an F-pair such that $a, b \notin D_2$, $|P_F^*(a,b)| \geq 5$, and let w be one of its inner vertices at distance at least three from both a and b. If none of the rules between Rule 2 and Rule 5 are applicable, then* $\deg_G(w) = \deg_{T(F)}(w)$.

Proof of Lemma 1. Suppose that this is not the case, and let $v \in N_G(w) \setminus N_{T(F)}(w)$.

- If $v \in P_F(a,b)$, suppose w.l.o.g. that w F-links v to a; moreover, let w_2 be the unique neighbor of v that F-links it to w. In this case, Rule 2 is applicable: a, v are an F-pair with the required properties, w_2 has the same role here as in the definition of the rule, and we may set w as w_1.
- If $v \notin \mathsf{leaves}\,(T(F))$, we may assume, w.l.o.g., that w F-links v to b. We can apply Rule 3: v is not adjacent to w in $T(F)$, so w has one neighbor $w_2 \in D_2$ that F-links it to v.
- If $v \in \mathsf{leaves}\,(T(F))$ and its unique neighbor is $w_2 \in D_2 \setminus P_F(a,b)$, we again may assume w.l.o.g. that w_2 F-links a and v. In this case, Rule 4 is applicable: there is some $z \notin \mathsf{leaves}\,(T(F))$ (possibly $z \in \{a, w_2\}$) that forms an F-pair with v, where $P_F(z,v) \setminus \{z, v\}$ may be empty if $z = w_2$.
- If $v \in \mathsf{leaves}\,(T(F))$ and $z \in D_*$ is its unique neighbor in $T(F)$, then, since there are at least two other vertices between w and each of the endpoints of $P_F(a,b)$, Rule 5 is applicable; to see that this is the case, set w to w_3 in the definition of the rule and w_1, w_2 as appropriate to depending on which endpoint of $P_F(a,b)$ F-links w to v.

Thus, we conclude that v cannot exist and that the statement holds. □

Theorem (8). *When parameterized by the size q of a feedback edge set,* Induced Tree *admits a kernel with $16q$ vertices and $17q$ edges that can be computed in $\mathcal{O}(q^2 + qn)$ time.*

Proof of Theorem 8. Let G be our n-vertex input graph and K a set of terminals. We begin by applying Rule 1 until no degree one vertex remains in G. Then, we take any feedback edge set F of G – we can obtain one in $\mathcal{O}(n)$ time by listing the set of back edges of a depth-first search tree of G – and construct $\mathcal{P}_F$ in $\mathcal{O}(n)$ time.

Let $\mathcal{P}_F$ be the set of paths between all F-pairs such that, for each $P_F(u,f) \in \mathcal{P}_F$ it holds that $u, f \notin D_2$. By Observation 1, we have at most $2q$ leaves in $T(F)$ and $2q$ vertices in D_*, so there are at most $4q$ paths in $\mathcal{P}_F$.

Each iteration of the first part of the algorithm is described below. If there is some path $P_F(u,f) \in \mathcal{P}_F$ with $f \in \mathsf{leaves}\,(T(F))$ and $P_F^*(u,f) \neq \emptyset$, check if there is an edge in F between f and one of the vertices of $P_F^*(u,f) \cup \{u\}$; if this is the case, apply Rule 2 in $\mathcal{O}(n)$ time and move on to the next iteration; by Observation 1, $|\mathcal{P}_F| \leq 4q$, so we can inspect each path and perform this step in $\mathcal{O}(q+n)$ time. Otherwise, let $vw_1 \in F$ be such that $w_1 \notin \mathsf{leaves}\,(T(F))$, and $w_1 \in P_F(u,f) \in \mathcal{P}_\mathcal{F}$. If $v \notin \mathsf{leaves}\,(T(F))$ and w_1 is adjacent to some $w_2 \in P_F^*(u,f)$

that F-links to v, apply Rule 3; we can check if these conditions are satisfied in $\mathcal{O}(n)$ time, in particular, F-linking is a matter of testing if w_1 and v are in the same connected component of $T(F) \setminus \{w_2\}$. If, however, $v \in \mathsf{leaves}\,(T(F)) \setminus \{f\}$, v forms an F-pair with $z \in V(G)$, $w_2 \in D_2 \cap N_{T(F)}(v)$, and u F-links f to z, Rule 4 is applicable in $\mathcal{O}(n)$ time. Finally, if $v \in \mathsf{leaves}\,(T(F)) \setminus \{f\}$ is adjacent to some $z \in D_*$ which F-links u and v, $w_3 \in P_F(u, f) \cap N_{T(F)}(u)$ is adjacent to w_2 which in turn is F-linked to f by w_1, Rule 5 is applicable is $\mathcal{O}(n)$ time.

If none of the conditions stated in the previous paragraph is satisfied, we stop the algorithm: the number of leaves of $T(F)$ cannot be increased by single edge swaps. As to the number of iterations, rules 2 through 5 guarantee that, when applicable, the number of leaves increases by exactly one. Since each one of their applications can be performed in $\mathcal{O}(q + n)$ time and we have at most $\max_F |\mathsf{leaves}\,(T(F))| \leq 2q$ iterations, the above algorithm can be executed in $\mathcal{O}\big(q^2 + qn\big)$ time.

Let $\mathcal{P}_\alpha$ be the set of paths of $\mathcal{P}_F$ whose endpoints are strict F-pairs. For each path $P_F(u, f)$ in $\mathcal{P}_F \setminus \mathcal{P}_\alpha$, let $u', f' \in P_F(u, f)$ be the strict F-pair that maximizes $|P_F(u', f')|$. Note that $|P_F(u, f) \setminus P_F(u', f')| \leq 4$ if there are leaves adjacent to each of the vertices at distance two from the endpoints; we refer to Figure 8 for an example. Finally, define $\mathcal{P}_\beta$ as $P_F(u', f')$ for each $P_F(u, f) \in \mathcal{P}_F \setminus \mathcal{P}_\alpha$.

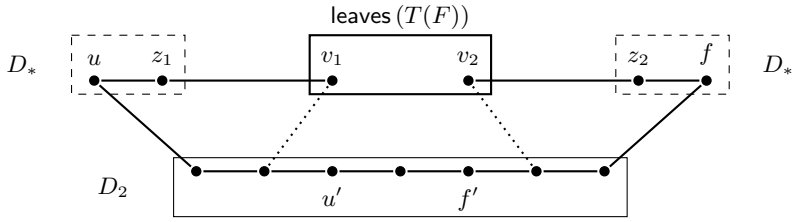

Fig. 8. Example of a path $P_F(u, f)$ and its longest subpath between a strict F-pair u', f'; dotted edges belong to F.

Now, for each path in $\mathcal{P}_\alpha \cup \mathcal{P}_\beta$, we can apply Rule 6; since at each step we remove one vertex from G, all applications of the rule amount to $\mathcal{O}(n)$-time. Afterwards, we apply Rule 7 to compress the paths as much as possible; again, this entire process is feasibly done in $\mathcal{O}(n)$ steps. As such, each path in $\mathcal{P}_\alpha \cup \mathcal{P}_\beta$ has size at most three; consequently each path in $\mathcal{P}_F$ has size at most seven. At first glance, this would yield a kernel of size $4q + 7 \cdot 4q = 32q$; however, we can observe that, for each edge in F incident to a vertex in some path of $\mathcal{P}_\beta$, we are essentially *reducing the number of leaves of* $T(F)$ *and large degree vertices in one unit each*: the bound of $2q$ leaves is only met with equality if *every* edge of F is incident to two leaves of $T(F)$. Therefore, if $\beta = |\mathcal{P}_\beta|$, the kernel's size is given by $|\mathsf{leaves}\,(T(F))| + |D_*| + 3|\mathcal{P}_\alpha| + 7|\mathcal{P}_\beta| \leq (2q - \beta) + (2q - \beta) + 3(4q - 2\beta) + 7\beta = 16q - \beta$, which is maximized when $\beta = 0$. Regarding the number of edges, the contracted graph has $16q$ vertices and a feedback edge set of size q, so it has at

most $17q$ edges. Finally, since the first part of the algorithm runs in $\mathcal{O}\big(q^2 + qn\big)$ time and the latter in $\mathcal{O}(n)$ time, we have a total complexity of $\mathcal{O}\big(q^2 + qn\big)$ time. □