When Maximum Stable Set can be solved in FPT time

Édouard Bonnet
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France
edouard.bonnet@ens-lyon.fr

Nicolas Bousquet
CNRS, G-SCOP laboratory, Grenoble-INP, France
nicolas.bousquet@grenoble-inp.fr

Stéphan Thomassé
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France
Institut Universitaire de France
stephan.thomasse@ens-lyon.fr

Rémi Watrigant
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France
remi.watrigant@ens-lyon.fr

Abstract

Maximum Independent Set (MIS for short) is in general graphs the paradigmatic $W[1]$-hard problem. In stark contrast, polynomial-time algorithms are known when the inputs are restricted to structured graph classes such as, for instance, perfect graphs (which includes bipartite graphs, chordal graphs, co-graphs, etc.) or claw-free graphs. In this paper, we introduce some variants of co-graphs with *parameterized noise*, that is, graphs that can be made into disjoint unions or complete sums by the removal of a certain number of vertices and the addition/deletion of a certain number of edges per incident vertex, both controlled by the parameter. We give a series of FPT Turing-reductions on these classes and use them to make some progress on the parameterized complexity of MIS in H-free graphs. We show that for every fixed $t \geqslant 1$, MIS is FPT in $P(1,t,t,t)$-free graphs, where $P(1,t,t,t)$ is the graph obtained by substituting all the vertices of a four-vertex path but one end of the path by cliques of size t. We also provide randomized FPT algorithms in dart-free graphs and in cricket-free graphs. This settles the FPT/W[1]-hard dichotomy for five-vertex graphs H.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Fixed parameter tractability

Keywords and phrases Parameterized Algorithms, Independent Set, H-Free Graphs

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

A *stable set* or *independent set* in a graph is a subset of vertices which are pairwise non-adjacent. Finding an independent set of maximum cardinality, called Maximum Independent Set (or MIS for short), is not only NP-hard to solve [18] but also to approximate within ratio $n^{1-\varepsilon}$ [23, 38]. One can then wonder if efficient algorithms exist with the additional guarantee that k, the size of the maximum stable set, is fairly small compared to n, the number of vertices of the input (think, for instance, $k \leqslant \log n$). It turns out that, for any computable function $h = \omega(1)$ (but whose growth can be arbitrarily slow), MIS is unlikely to admit a polynomial-time algorithm even when $k \leqslant h(n)$. In parameterized complexity terms, MIS is $W[1]$-hard [16]. More quantitatively, MIS cannot be solved in time $f(k)n^{o(k)}$ for any computable function f, unless the Exponential Time Hypothesis fails. This is quite a statement when a trivial algorithm for MIS runs in time n^{k+2}, and a simple reduction to triangle detection yields a $n^{\frac{\omega k}{3}+O(1)}$-algorithm, where ω is the best exponent known for matrix multiplication.

It thus appears that MIS on general graphs is totally impenetrable. This explains why efforts have been made on solving MIS in subclasses of graphs. The most emblematic result in that line of works is a polynomial-time algorithm in perfect graphs [20]. Indeed, perfect graphs generalize many graph classes for which MIS is in P: bipartite graphs, chordal graphs, co-graphs, etc. In this paper, we put the focus on classes of graphs for which MIS can be solved in FPT time (rather than in polynomial-time). For graphs with bounded degree Δ, the simple branching algorithm has FPT running time $(\Delta+1)^k n^{O(1)}$. The same observation also works more generally for graphs with bounded average degree, or even d-degenerate graphs. A non-trivial result is that MIS remains FPT in arguably the most general class of sparse graphs, *nowhere dense graphs*. Actually, deciding first-order formulas of size k can be done in time $f(k)n^{1+\varepsilon}$ on any nowhere dense class of graphs [19]. Since MIS and the complement problem, Maximum Clique, are both expressible by a first-order formula of length $O(k^2)$, $\exists v_1, \ldots, v_k \bigwedge_{i,j} (\neg) E(v_i, v_j)$, there is an FPT algorithm on nowhere dense graphs and also on complements of nowhere dense graphs. A starting point of the present paper is to design FPT Turing-reductions in classes containing both very dense and very sparse graphs.

Co-graphs with parameterized noise. If G and H are two graphs, we can define two new graphs: $G \cup H$, their disjoint union, and $G \oplus H$ their (complete) sum, obtained from the disjoint union by adding all the edges from a vertex of G to a vertex of H. Then, the hereditary class of co-graphs can be inductively defined by: K_1 (an isolated vertex) is a co-graph, and $G \cup H$ and $G \oplus H$ are co-graphs, if G and H are themselves co-graphs. So the construction of a co-graph can be seen as a binary tree whose internal nodes are labeled by $\cup$ or $\oplus$, and leaves are K_1. Finding the tree of operations building a given co-graph, sometimes called *co-tree*, can be done in linear time [10]. This gives a simple algorithm to solve MIS on co-graphs: $\alpha(K_1) = 1$, $\alpha(G \cup H) = \alpha(G) + \alpha(H)$, and $\alpha(G \oplus H) = \max(\alpha(G), \alpha(H))$.

We add a *parameterized noise* to the notion of co-graphs. More precisely, we consider graphs that can be made disjoint unions or complete sums by the deletion of $g(k)$ vertices and the edition (*i.e.*, addition or deletion) of $d(k)$ edges per incident vertex. We design a series of FPT Turing-reductions on several variants of these classes using the so-called *iterative expansion* technique [9, 4], Cauchy-Schwarz-like inequalities, and Kővári-Sós-Turán's theorem. This serves as a crucial foundation for the next part of the paper, where we explore the parameterized complexity of MIS in H-free graphs (*i.e.*, graphs not containing H as an induced subgraph). However, we think that the FPT routines developed on *co-graphs with parameterized noise* may also turn out to be useful outside the realm of H-free graphs.

Classical and parameterized dichotomies in H-free graphs. The question of whether MIS is in P or NP-complete in H-free graphs, for each connected graph H, goes back to the early eighties. However, a full dichotomy is neither known nor does it seem within reach in the near future. For three positive integers i, j, k, let $S_{i,j,k}$ be the tree with exactly one vertex of degree three, from which start three paths with i, j, and k edges, respectively. The claw is the graph $S_{1,1,1}$, thus $\{S_{i,j,k}\}_{i \leqslant j \leqslant k}$ is the set of all the subdivided claws. We denote by P_ℓ the path on ℓ vertices.

If G' is the graph obtained by subdividing each edge of a graph G exactly $2t$ times, Alekseev observed that $\alpha(G') = \alpha(G) + t|E(G)|$ [1]. This shows that MIS remains NP-hard on graphs which locally look like paths or subdivided claws (one can perform the subdivision on sub-cubic graphs G, on which MIS remains NP-complete). In other words, if a connected graph H is not a path nor a subdivided claw then MIS is NP-complete on H-free graphs [1]. The MIS problem is easy on P_4-free graphs, which are exactly the co-graphs. Already on P_5-free graphs, a polynomial algorithm is much more difficult to obtain. This was done by Lokshtanov *et al.* [27] using the framework of potential maximal cliques. A quasi-polynomial algorithm was proposed for P_6-free graphs [26], and recently, a polynomial-time algorithm was found by Grzesik *et al.* [21]. Brandstädt and Mosca showed how to solve MIS in polynomial-time on (P_7, triangle)-free graphs [7]. This result was then generalized by the same authors on ($S_{1,2,4}$, triangle)-free graphs [8], and by Maffray and Pastor on (P_7, bull[1])-free graphs (as well as ($S_{1,2,3}$, bull)-free graphs)

[1] the bull is obtained by adding a pendant neighbor to two distinct vertices of the triangle (K_3)

[32]. Bacsó *et al.* [3] presented a subexponential-time $2^{O(\sqrt{tn \log n})}$-algorithm in P_t-free graphs, for every integer t. Nevertheless, the classical complexity of MIS remains wide open on P_t-free graphs, for $t \geqslant 7$.

On claw-free graphs MIS is known to be polynomial-time solvable [35, 36]. Recently, this result was generalized to ℓclaw-free graphs [6] (where ℓclaw is the disjoint union of ℓ claws). On fork-free graphs (the fork is $S_{1,1,2}$) MIS admits a polynomial-time algorithm [2], and so does its weighted variant [30]. The complexity of MIS is open for $S_{1,1,3}$-free graphs and $S_{1,2,2}$-free graphs, and there is no triple $i \leqslant j \leqslant k$, for which we know that MIS is NP-hard on $S_{i,j,k}$-free graphs. Some subclasses of $S_{i,j,k}$-free graphs are known to admit polynomial algorithms for MIS: for instance $(S_{1,1,3}, K_{t,t})$-free graphs [14], subcubic $S_{2,t,t}$-free graphs [22] (building upon [31], and generalizing results presented in [33, 34] for subcubic *planar* graphs), bounded-degree $tS_{1,t,t}$-free graphs [29], for any fixed positive integer t. This leads to the following conjecture:

▶ **Classical MIS Dichotomy Conjecture(H).** *For every connected graph H,* Maximum Independent Set *in H-free graphs is in P iff $H \in \{P_\ell\}_\ell \cup \{S_{i,j,k}\}_{i \leqslant j \leqslant k}$.*

An even stronger conjecture is postulated by Lozin (see Conjecture 1 in [28]). Dabrowski *et al.* initiated a systematic study of the parameterized complexity of MIS on H-free graphs [12, 13]. In a nutshell, parameterized complexity aims to design $f(k)n^{O(1)}$-algorithms (FPT algorithm, for Fixed-Parameter Tractable), where n is the size of the input, and k is the size of the solution (or another well-chosen parameter), for most often NP-hard problems. The so-called W-hierarchy (and in particular, $W[1]$-hardness) and the Exponential Time Hypothesis (ETH) both provide a framework to rule out such a running time. We refer the interested reader to two recent textbooks [16, 11] and to a survey on the ETH and its consequences [25]. In the language of parameterized complexity, the dichotomy problem is the following:

▶ **Parameterized MIS Dichotomy(H).** *Is* MIS *(randomized) FPT or $W[1]$-hard in H-free graphs?*

This question may be even more challenging than its classical counterpart. Indeed, there is no FPT algorithm known for the classical open cases: P_7-, $S_{1,1,3}$-, and $S_{1,2,2}$-free graphs. Besides, the reduction of Alekseev [1] that we mentioned above does not show $W[1]$-hardness. Thus, there are *a priori* more candidates H for which the parameterized status of MIS is open. For instance, by Ramsey's theorem, MIS is FPT on K_t-free graphs, for any fixed t. Observe that a randomized FPT algorithm for a W[1]-hard problem is highly unlikely, as it would imply a randomized algorithm solving 3-SAT in subexponential time.

Dabrowski *et al.* showed that MIS is FPT[2] in H-free graphs, for all H on four vertices, except $H = C_4$ (the cycle on four vertices). Thomassé *et al.* presented an FPT algorithm on bull-free graphs [37], whose running time was later improved by Perret du Cray and Sau [17]. Bonnet *et al.* provided three variants of a parameterized counterpart of Alekseev's reduction [4, 5]. Although the description of the open cases (see Figure 1) is not nearly as nice and compact as for the classical dichotomy, it is noteworthy that they almost correspond to paths and subdivided claws where vertices are blown-up into cliques.

Let us make that idea more formal. Substituting a graph H at a vertex v of a graph G gives a new graph with vertex set $(V(G) \setminus \{v\}) \cup V(H)$, and the same edges as in G and H, plus all the edges xy where $x \in V(H)$, $y \in V(G)$, and $vy \in E(G)$. For a sequence of positive integers $a_1, a_2, \ldots, a_\ell$, we denote by $P(a_1, a_2, \ldots, a_\ell)$ the graph obtained by substituting a clique K_{a_i} at the i-th vertex of a path P_ℓ, for every $i \in [\ell]$. We also denote by $T(a, b, c)$ the graph obtained by substituting a clique K_a, K_b, and K_c to the first, second, and third leaves, respectively, of a claw. Thus, $T(1, 1, 1)$ is the claw and $T(1, 1, 2)$ is called the cricket (see Figure 3d).

[2] here and in what follows, the parameter is the size of the solution

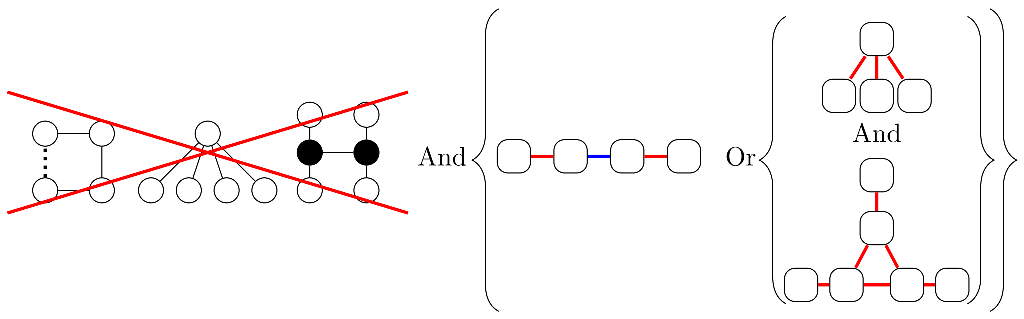


Figure 1 The dotted edge represents a path with at least one edge. The filled vertices emphasize two vertices with degree at least three in a tree. The rounded boxes are cliques. A red edge corresponds to a complete bipartite minus at most one edge. A blue edge correspond to a $2K_2$-free bipartite graph. The FPT connected candidates H have to be chordal, without induced $K_{1,4}$ or trees with two branching vertices (*i.e.*, vertices of degree at least 3), and have to fit on a path with at most one blue edge (and the rest of red edges) or both in a subdivided claw and a line-graph of a subdivided claw with red edges only. A further restriction in the line-graph of subdivided claw is that three vertices each in a different clique of the triangle of red edges cannot induce a $K_1 \cup K_2$ (see [4]).

We show in this paper that MIS is (randomized) FPT in $T(1,1,2)$-free graphs (or cricket-free graphs). This is in sharp contrast with the $W[1]$-hardness for $T(1,2,2)$-free graphs [5] (see Figure 2e). It disproves a seemingly reasonable conjecture that FPTness is preserved by adding a true twin to a vertex of H. We thus have a fairly good understanding of the parameterized complexity of MIS when H is obtained by substituting cliques on a claw. We therefore turn towards the graphs H obtained by substituting cliques on a path. MIS was shown FPT on $P(t,t,t)$-free graphs [4]. A natural next step is to attack the following conjecture.

▶ **Conjecture 1.** *For any integer t,* MIS *can be solved in FPT time in $P(t,t,t,t)$-free graphs.*

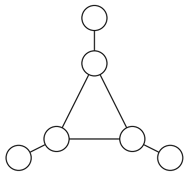
We denote by $P_\ell(t)$ the graph $P(t,t,\ldots,t)$ where the sequence $t,t,\ldots,t$ is of length ℓ. We further conjecture the following, which is a far more distant milestone.

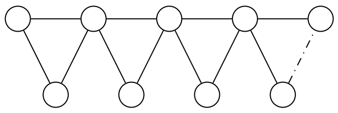
▶ **Conjecture 2.** *For any integers t and ℓ,* MIS *is FPT in $P_\ell(t)$-free graphs.*

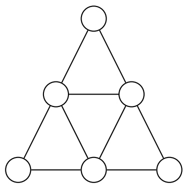
Let us recall though that the parameterized complexity of MIS is open in P_7-free graphs, and no *easy* FPT algorithm is known on P_5-free graphs. In general, we believe that there will be very few connected candidates (as described by Figure 1) which will not end up in (randomized) FPT. As a first empirical evidence, we show that the four candidates remaining among the 34 graphs on five vertices indeed all lead to (randomized) FPT algorithms.

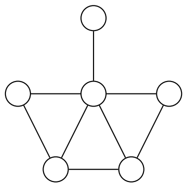
Organization of our results. The rest of the paper is organized as follows. In Section 2, we introduce FPT Turing-reductions relevant to the subsequent section. In Section 3, we give a series of FPT algorithms in far-reaching generalizations of co-graphs: graphs where the deletion of $g(k)$ vertices leads to a separation which is either very sparse or very dense, in a way that is controlled by the parameter. In Section 4, we use these results to obtain an FPT algorithm on $P(1,t,t,t)$-free graphs for any positive integer t, taking a stab at Conjecture 1. Observe that this result settles at the same time $\overline{P}$ ($=P(1,1,1,2)$) and the kite ($=P(1,1,2,1)$). In Section 5, we finish the FPT/$W[1]$-hard classification for five-vertex graphs by designing randomized FPT algorithms on dart-free graphs and cricket-free graphs.

We believe that the results of Section 3 as well as the techniques developed in Sections 4 and 5 may help in settling Conjecture 1. For $P(t,t,t,t,t)$-free graphs, it is possible that one will have to combine the framework of potential maximal cliques with our techniques. To solve Conjecture 2, let alone the full parameterized dichotomy, some new ideas will be needed. The FPT algorithms of the current paper


(a) The net.


(b) The chain of four triangles with or without the dash-dotted edge.


(c) The triforce.


(d) $\overline{\mathrm{Gem} \cup K_1}$.

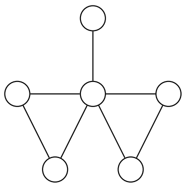
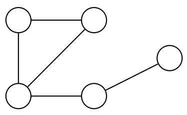
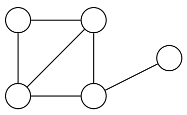
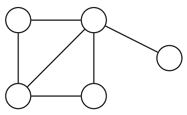

(e) $T_{1,2,2} = \overline{W_4 \cup K_1}$.

Figure 2 Some connected chordal $K_{1,4}$-free graphs H for which H-free MIS is $W[1]$-hard (see [4]). These graphs do not fit the candidate forms of Figure 1 for subtle reasons and illustrate how delicate the parameterized dichotomy promises to be. In particular, observe that MIS is $W[1]$-hard on $T(1, 2, 2)$-graphs, whereas we will show in this paper that it is FPT in $T(1, 1, 2)$-graphs (a.k.a. cricket-free graphs).


(a) $\bar{P}$


(b) Kite


(c) Dart

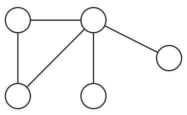

(d) Cricket

Figure 3 The four (out of 34) remaining cases on five vertices for the FPT/$W[1]$-hard dichotomy (see [5]). In this paper, we come up with new tools and solve all of them in (randomized) FPT.

merely serve for classification purposes, and are not practical. A possible line of work is to get improved running times for the already established FPT cases. We also hope that the results of Section 3 will prove useful in a context other than H-free graphs.

2 Preliminaries

Here, we introduce some basics about graph notations, Ramsey numbers, and FPT algorithms.

2.1 Notations

For any pair of integers $i \leqslant j$, we denote by $[i, j]$ the set of integers $\{i, i+1, \ldots, j-1, j\}$, and for any positive integer i, $[i]$ is a shorthand for $[1, i]$. We use the standard graph terminology and notations [15]. All our graphs are finite and simple, *i.e.*, they have no multiple edge nor self-loop. For a vertex v, we denote by $N(v)$ the set of neighbors of v, and $N[v] := N(v) \cup \{v\}$. For a subset of vertices S, we set $N(S) := \bigcup_{v \in S} N(v) \setminus S$ and $N[S] := N(S) \cup S$. The *degree* (resp. *co-degree*) of a vertex v is $|N(v)|$ (resp. $|V \setminus N[v]|$). If G is a graph and X is a subset of its vertices, $G[X]$ is the subgraph induced by X and $G - X$ is a shorthand for $G[V(G) \setminus X]$. We denote by $\alpha(G)$ the independence number, that is the size of a maximum independent set. If H and G are two graphs, we write $H \subseteq_i G$ to mean that H is an induced subgraph of G, and $H \subset_i G$ if H is a proper induced subgraph of G. We denote by K_ℓ, P_ℓ, C_ℓ, the clique, path, cycle, respectively, on ℓ vertices, and by $K_{s,t}$ the complete bipartite graph with s vertices on one side and t, on the other. The claw is $K_{1,3}$, and the paw is the graph obtained by adding one edge to the claw. If H is a graph and t is a positive integer, we denote by tH the graph made of t disjoint copies of

H. For instance, $2K_2$ corresponds to the disjoint union of two edges. We say that a class of graphs $\mathcal{C}$ is *hereditary* if it is closed by induced subgraph, *i.e.*, $\forall H, G$, $(G \in \mathcal{C} \wedge H \subseteq_i G) \Rightarrow H \in \mathcal{C}$.

2.2 Ramsey numbers

For two positive integers a and b, $R(a, b)$ is the smallest integer such that any graph with at least that many vertices has an independent set of size a or a clique of size b. By Ramsey's theorem, $R(a, b)$ always exists and is no greater than $\binom{a+b}{a}$. For the sake of convenience, we set $Ram(a, b) := \binom{a+b}{a} = \binom{a+b}{b}$. We will use repeatedly a constructive version of Ramsey's theorem.

▶ **Lemma 3** (folklore). *Let a and b be two positive integers, and let G be a graph on at least $Ram(a, b)$ vertices. Then an independent set of size a or a clique of size b can be found in linear time.*

Proof. We show this lemma by induction on $a + b$. For $a = 1$ (or $b = 1$), any vertex of G works (it is a clique and an independent set at the same time). And G is non-empty since it has at least $\binom{1+b}{1}$ (or $\binom{a+1}{1}$) vertices. We assume $a, b \geqslant 2$ and consider any vertex v of G. Let $G_1 := G - N[v]$ and $G_2 := G[N(v)]$, so $|V(G)| = 1 + |V(G_1)| + |V(G_2)|$.

Since $|V(G)| \geqslant \binom{a+b}{a} = \binom{a+b-1}{a-1} + \binom{a+b-1}{a}$, it cannot be that both $|V(G_1)| \leqslant Ram(a-1, b) - 1$ and $|V(G_2)| \leqslant Ram(a, b-1) - 1$. If G_1 has at least $Ram(a-1, b)$ vertices, we find by induction an independent set I of size $a - 1$ or a clique of size b. Thus $I \cup \{v\}$ is an independent set of size a in G. If instead G_2 has at least $Ram(a, b-1)$ vertices, we find by induction an independent set of size a or a clique C of size $b - 1$. Thus $C \cup \{v\}$ is an independent set of size b in G. ◀

For two positive integers a and b, we denote by $Ram_a(b)$ the smallest integer n such that any edge-coloring of K_n with a colors has a monochromatic clique of size b. In particular, $Ram_2(b) = Ram(b, b)$ (one color for the edges, and one color for the non-edges). Again, $Ram_a(b)$ always exists and a monochromatic clique of size b in an a-edge-colored clique of size at least $Ram_a(b)$ can be found in polynomial-time (whose exponent does *not* depend on a and b).

2.3 FPT Turing-reductions

For an instance (I, k) of MIS, let yes(I, k) be the Boolean function which equals $True$ if and only if (I, k) is a positive instance.

▶ **Definition 4.** *A* decreasing FPT g-Turing-reduction *is an FPT algorithm which, given an instance (I, k), produces $\ell := g(k)$ instances $(I_1, k_1), \dots, (I_\ell, k_\ell)$, for some computable function g, such that:*

- *(i) $yes(I, k) \Leftrightarrow \phi(yes(I_1, k_1), \dots, yes(I_\ell, k_\ell))$, where ϕ is a fixed FPT-time checkable formula*[3],
- *(ii) $|I_j| \leqslant |I|$ for every $j \in [\ell]$, and*
- *(iii) $k_j \leqslant k - 1$ for every $j \in [\ell]$.*

Note that conditions (ii) and (iii) prevent the instance size from increasing and force the parameter to strictly decrease, respectively.

▶ **Lemma 5.** *Assume there is a decreasing FPT g-Turing-reduction for* MIS *on every input $(G \in \mathcal{C}, k)$, running in time $h(k)|V(G)|^\gamma$ (this includes the time to check ϕ). Let $f : [k-1] \to \mathbb{N}$ be a non-decreasing function. If any instance (H, k') with $k' < k$ can be solved in time $f(k')|V(H)|^c$ with $c \geqslant \gamma$, then* MIS *can be solved in FPT time $f(k)|V(G)|^c$ in $\mathcal{C}$, with $f(k) := h(k) + g(k)f(k-1)$.*

[3] By *FPT-time checkable formula*, we mean that there exists an algorithm which takes as input ℓ Booleans $b_1, \dots, b_\ell$ and tests whether $\phi(b_1, \dots, b_\ell)$ is true in FPT time parameterized by ℓ.

Proof. We show the lemma by induction. If $k = 1$, this is immediate. We therefore assume that $k \geqslant 2$. We apply the decreasing FPT g-Turing-reduction to (G, k). That creates at most $g(k)$ instances with parameter at most $k - 1$. We solve each instance in time $f(k-1)n^c$ with $n := |V(G)|$. The overall running time is bounded by $h(k)n^\gamma + g(k)f(k-1)n^c \leqslant f(k)n^c$ by extending the partial function f with $f(k) := h(k) + g(k)f(k-1)$. ◀

This corollary follows by induction on the parameter k.

▶ **Corollary 6.** *If* MIS *admits a decreasing FPT g-Turing-reduction on a hereditary class, then* MIS *can be solved in FPT time in $\mathcal{C}$.*

▶ **Definition 7.** *An* improving FPT g-Turing-reduction *is an FPT time $h(k)|V(G)|^\gamma$ algorithm which, given an instance (I, k), produces some instances $(I_1, k_1), \dots, (I_\ell, k_\ell)$, and can check a formula ϕ, such that:*

- *(i) $yes(I, k) \Leftrightarrow \phi(yes(I_1, k_1), \dots, yes(I_\ell, k_\ell))$, and*
- *(ii) $\exists c_0, f_0, \forall c \geqslant c_0, f \in \Omega(f_0)$, $h(k)|V(G)|^\gamma + \sum_{j \in [\ell]} f(k_j)|I_j|^c \leqslant f(k)|I|^c$.*

▶ **Lemma 8.** *Assume there is an improving FPT g-Turing-reduction for* MIS *on every input $(I \in \mathcal{C}, k)$, producing in time $h(k)|I|^\gamma$, some instances $(I_1, k_1), \dots, (I_\ell, k_\ell)$. If each instance (I_j, k_j) can be solved in time $h(k_j)|I_j|^{c'}$, then* MIS *can be solved in FPT time in $\mathcal{C}$.*

Proof. Let $c := \max(c_0, c')$ and $f := \max(f_0, h)$, for the c_0 and f_0 of Definition 8. *A fortiori*, instances (I_j, k_j) can be solved in time $f(k_j)|I_j|^c$. We call the Turing-reduction on (I, k), solve every subinstances (I_j, k_j), and check ϕ. By item (ii), the overall running time $h(k)|V(G)|^\gamma + \sum_{j \in [\ell]} f(k_j)|I_j|^c$ is bounded by $f(k)|I|^c$. By item (i), this decides (I, k). ◀

When trying to compute MIS in FPT time, one can assume that there is no vertex of bounded degree or bounded co-degree (in terms of a function of k).

▶ **Observation 9.** *Let (G, k) be an input of* MIS *with a vertex v of degree $g(k)$ for some computable function g. Then the instance admits a decreasing FPT Turing-reduction.*

Proof. A maximal independent set has to intersect $N[v]$. So, we can branch on $g(k) + 1$ instances with parameter $k - 1$. ◀

▶ **Observation 10.** *Let (G, k) be an input of* MIS *with a vertex v of co-degree $g(k)$ for some computable function g. Then the instance admits an improving FPT Turing-reduction.*

Proof. We can find the vertex v in time $ng(k)$ with $n := |V(G)|$, and we assume $n \geqslant 2$. By branching on v, we define two instances $(G - N[v], k-1)$ and $(G - \{v\}, k)$ (which corresponds to including v to the solution, or not). The first instance can be further reduced in time $g(k)^{k-1}$ (by actually solving it). So the two instances output by the Turing-reduction are Bool and $(G - \{v\}, k)$, where *Bool* is the result of solving $(G - N[v], k-1)$. The formula ϕ is just Bool$\vee$yes$(G - \{v\}, k)$. Let $c_0 := 2$ and $f_0(k) := g(k)^{k-1}$. For all $c \geqslant c_0$ and $f \in \Omega(f_0)$, $ng(k) + g(k)^{k-1} + f(k)(n-1)^c \leqslant nf(k) + f(k) + f(k)(n-1)^c \leqslant f(k)(n + 1 + (n-1)^c) \leqslant f(k)n^c$. ◀

3 Almost disconnected and almost join graphs

We say that a graph is a join or a *complete sum*, if there is a non-trivial bipartition (A, B) of its vertex set (*i.e.* A and B are non-empty) such that every pair of vertices $(u, v) \in A \times B$ is linked by an edge. Equivalently, a graph is a complete sum if its complement is disconnected. In the following subsection, we define a series of variants of complete sums and disjoint unions in the presence of a *parameterized noise*.

3.1 Definition of the classes

In all the following definitions, we say that a tripartition (A, B, R) is *non-trivial* if A and B are non-empty and $|R| < \min(|A|, |B|)$. Notice that we do not assume R is non-empty.

▶ **Definition 11.** *Graphs in a class $\mathcal{C}$ are* (g, d)-almost disconnected *if there exist two computable functions g and d, such that for every $G \in \mathcal{C}$ and $k \geqslant \alpha(G)$, there is a non-trivial tripartition (A, B, R) of $V(G)$ satisfying:*

- $|R| \leqslant g(k)$, *and*
- $\forall v \in A$, $|N(v) \cap B| \leqslant d(k)$ *and* $\forall v \in B$, $|N(v) \cap A| \leqslant d(k)$.

Similarly, we define a generalization of a complete sum.

▶ **Definition 12.** *Graphs in a class $\mathcal{C}$ are* (g, d)-almost bicomplete *if there exist two computable functions g and d, such that for every $G \in \mathcal{C}$ and $k \geqslant \alpha(G)$, there is a non-trivial tripartition (A, B, R) of $V(G)$ satisfying:*

- $|R| \leqslant g(k)$, *and*
- $\forall v \in A$, $|B \setminus N(v)| \leqslant d(k)$ *and* $\forall v \in B$, $|A \setminus N(v)| \leqslant d(k)$.

By extension, if $\mathcal{C}$ only contains graphs which are almost disconnected (resp. (g, d)-almost disconnected, almost bicomplete, (g, d)-almost bicomplete), then we say that $\mathcal{C}$ is almost disconnected (resp. (g, d)-almost disconnected, almost bicomplete, (g, d)-almost bicomplete). Note that we do not require an almost disconnected or an almost bicomplete class to be hereditary. For $G \in \mathcal{C}$, we call a satisfying tripartition (A, B, R) a *witness of almost disconnectedness* (resp. *witness of almost bicompleteness*).

We define the one-sided variants.

▶ **Definition 13.** *Graphs in a class $\mathcal{C}$ are* one-sided (g, d)-almost disconnected *if there exist two computable functions g and d, such that for every $G \in \mathcal{C}$ and $k \geqslant \alpha(G)$, there is a non-trivial tripartition (A, B, R) of $V(G)$ satisfying:*

- $|R| \leqslant g(k)$,
- $|B| > kd(k)$, *and*
- $\forall v \in A$, $|N(v) \cap B| \leqslant d(k)$.

In the above definition, the second condition is purely a technical one. Observe, though, that any tripartition (A, B, R) with $|R| < |B| \leqslant d(k)$ trivially satisfies the third condition (provided $|R| < d(k)$). So a condition forcing B to have more than $d(k)$ vertices is somehow needed. Now, we set the lower bound on $|B|$ a bit higher to make Lemma 18 work. Similarly, we could define the one-sided generalization of a complete sum.

▶ **Definition 14.** *Graphs in a class $\mathcal{C}$ are* one-sided (g, d)-almost bicomplete *if there exist two computable functions g and d, such that for every $G \in \mathcal{C}$ and $k \geqslant \alpha(G)$, there is a non-trivial tripartition (A, B, R) of $V(G)$ satisfying:*

- $|R| \leqslant g(k)$,
- *if there is an independent set of size k, there is one that intersects A, and*
- $\forall v \in B$, $|A \setminus N(v)| \leqslant d(k)$.

Again, the second condition is there to make Theorem 20 work.

3.2 Improving and decreasing FPT Turing-reductions

The following technical lemma will be used to bound the running time of recursive calls on two almost disjoint parts of the input.

▶ **Lemma 15.** *Suppose $\gamma \geqslant 0$ and $c \geqslant \max(2, \gamma+2)$ are two constants, and n_1, n_2, n, u are four positive integers such that $n_1 + n_2 + u = n$ and $\min(n_1, n_2) > u$. Then,*

$$n^\gamma + (n_1+u)^c + (n_2+u)^c < n^c.$$

Proof. First we observe that $n^2-((n_1+u)^2+(n_2+u)^2) = n_1^2+n_2^2+u^2+2(n_1n_2+n_1u+n_2u)-(n_1^2+2n_1u+u^2+n_2^2+2n_2u+u^2) = 2n_1n_2-u^2 > 2u^2-u^2 = u^2 \geqslant 1$. Now, $n^c = n^{c-2}n^2 \geqslant n^{c-2}(1+(n_1+u)^2+(n_2+u)^2) \geqslant n^{c-2}(n^{\gamma-c+2}+(n_1+u)^2+(n_2+u)^2) = n^\gamma + n^{c-2}(n_1+u)^2 + n^{c-2}(n_2+u)^2 > n^\gamma + (n_1+u)^c + (n_2+u)^c$. The last inequality holds since $n > n_1+u$ and $n > n_2+u$. ◀

We start with an improving FPT Turing-reduction on almost bicomplete graphs. It finds a kernel for solutions intersecting both A and B, solves recursively on $A \cup R$ and $B \cup R$ for the other solutions, and uses Lemma 15 to bound the overall running time.

▶ **Lemma 16.** *Let $\mathcal{C}$ be a (g,d)-almost bicomplete class of graphs. Suppose for every $G \in \mathcal{C}$, a witness (A,B,R) of almost bicompleteness can be found in time $h(k)|V(G)|^\gamma$. Then,* MIS *admits an improving FPT Turing-reduction in $\mathcal{C}$. In particular,* MIS *can be solved in FPT time if both $(G[A \cup R], k)$ and $(G[B \cup R], k)$ can.*

Proof. We can detect a potential solution S intersecting both A and B in time $n^2(2d(k)+g(k))^k = n^2s(k)$, with $n := |V(G)|$, by setting $s(k) := (2d(k)+g(k))^{k-2}$. We exhaustively guess one vertex $a \in S \cap A$ and one vertex $b \in S \cap B$. For each of these quadratically many choices, there are at most $d(k)$ non-neighbors of a in B and at most $d(k)$ non-neighbors of b in A. So the remaining instance $G - (N(a) \cup N(b))$ has at most $2d(k)+g(k)$ vertices; hence the running time.

We are now left with potential solutions intersecting A but not B, or B but not A. These are fully contained in $A \cup R$ or in $B \cup R$. Let $n_1 := |A|$ and $n_2 := |B|$ (so $n = n_1+n_2+|R|$). The two last branches just consist of recursively solving the instances $(G[A \cup R], k)$ and $(G[B \cup R], k)$. Let $c_0 := \max(4, \gamma+2)$ and $f_0 := h + s$. For all $c \geqslant c_0$ and $f \in \Omega(f_0)$,

$$h(k)n^\gamma + s(k)n^2 + f(k)(n_1+g(k))^c + f(k)(n_2+g(k))^c$$

$$\leqslant f(k)n^{\max(\gamma,2)} + f(k)(n_1+g(k))^c + f(k)(n_2+g(k))^c \leqslant f(k)n^c.$$

The last inequality holds by Lemma 15, since $\max(\gamma,2)+2 \leqslant c$ and $\min(n_1,n_2) > g(k)$. The conclusion holds by Lemma 8. ◀

If we only have *one-sided* almost bicompleteness, we need some additional conditions on the solution: at least one solution should intersect A (see Definition 14). We recall that $H \subset_i G$ means that H is a proper induced subgraph of G.

▶ **Lemma 17.** *Let $\mathcal{C}$ be a one-sided (g,d)-almost bicomplete class of graphs. Suppose for every $G \in \mathcal{C}$, a witness (A,B,R) of one-sided almost bicompleteness can be found in time $h(k)|V(G)|^\gamma$. Then,* MIS *admits an improving FPT Turing-reduction in $\mathcal{C}$. In particular,* MIS *can be solved in FPT time if $(G[A \cup R], k)$ and $\forall k' \leqslant k-1$, $\forall H \subset_i G$, (H, k') all can.*

Proof. Let S be an unknown solution. Let $k_1 := S \cap A$ and $k_2 := S \cap B$. Let us anticipate on an FPT running time $f(k)n^c$ for instances of size n and parameter k (the definition of f will be given later). For instance, covering the case $k_2 = 0$ takes time $f(k)|A \cup R|^c$, since it consists in solving $(G[A \cup R], k)$. By assumption, we do not have to consider the case $k_1 = 0$. For each pair k_1, k_2 such that $k_1 \geqslant 1, k_2 \geqslant 1, k_1 + k_2 \leqslant k$, we do the following.

An independent set of size k_1 in $G[A]$ is *candidate* if it is in the non-neighborhood of at least one vertex $v \in B$. Since $k_2 \geqslant 1$, we can restrict the search in A to candidate independent sets of size k_1. Indeed, any independent set in A, not in the non-neighborhood of any vertex of B, cannot be extended to k_2 ($\geqslant 1$)

more vertices of B. For each candidate independent set I_1 of size k_1, we compute an independent set of size k_2 in $B \setminus N(I_1)$. This takes time

$$\sum_{\substack{I_1 \text{ candidate} \\ |I_1|=k_1}} f(k_2)|B \setminus N(I_1)|^c = f(k_2) \sum_{\substack{I_1 \text{ candidate} \\ |I_1|=k_1}} |B \setminus N(I_1)|^c \leqslant f(k_2) \left(\sum_{\substack{I_1 \text{ candidate} \\ |I_1|=k_1}} |B \setminus N(I_1)| \right)^c$$

by Cauchy-Schwarz inequality (since $c \geqslant 2$). Now, since $k_1 > 0$,

$$\sum_{\substack{I_1 \text{ candidate} \\ |I_1|=k_1}} |B \setminus N(I_1)| \leqslant \sum_{\substack{I_1 \text{ candidate} \\ |I_1|=k_1}} |I_1| \cdot |B \setminus N(I_1)| \leqslant \binom{d(k)}{k_1} d(k)|B|.$$

The last inequality holds since $\sum_{I_1 \text{ candidate},|I_1|=k_1} |I_1| \cdot |B \setminus N(I_1)|$ counts the number of non-edges between A and B with multiplicity at most $\binom{d(k)}{k_1}$. Indeed a same non-edge uv (with $u \in A$, $v \in B$) is counted for at most $\binom{d(k)}{k_1}$ candidate independent sets (since they have to be in the non-neighborhood of v). Since, by assumption, vertices in B have at most $d(k)$ non-neighbors in A, the total number of non-edges is $d(k)|B|$. Let $c_0 \geqslant \gamma + 2$ and $f_0 := \max(h, k \mapsto k^{2k} \binom{d(k)}{k}^{ck} d(k)^{ck})$. For any $c \geqslant c_0$ and $f \in \Omega(f_0)$,

$$h(k)|V(G)|^\gamma + f(k)|A \cup R|^c + \sum_{k_1 \in [k-1], k_2 \in [k-k_1]} f(k_2) \left(\sum_{\substack{I_1 \text{ candidate} \\ |I_1|=k_1}} |B \setminus N(I_1)| \right)^c$$

$$\leqslant h(k)|V(G)|^\gamma + f(k)|A \cup R|^c + k^2 f(k-1) \binom{d(k)}{k}^c d(k)^c |B|^c$$

$$\leqslant f(k)|V(G)|^\gamma + f(k)|A \cup R|^c + f(k)|B|^c \leqslant f(k)|V(G)|^c$$

since $f(k) \geqslant k^2 \binom{d(k)}{k}^c d(k)^c f(k-1)$. The last inequality holds by Lemma 15. The conclusion holds by Lemma 8. ◀

We now turn our attention to almost disconnected classes. For these classes, we obtain decreasing FPT Turing-reductions, *i.e.*, where the produced instances have a strictly smaller parameter than the original instance.

▶ **Lemma 18.** *Let $\mathcal{C}$ be a one-sided (g, d)-almost disconnected class of graphs. Suppose for every $G \in \mathcal{C}$, a witness (A, B, R) of one-sided almost disconnectedness can be found in time $h(k)|V(G)|^\gamma$. Then,* MIS *admits a decreasing FPT Turing-reduction in $\mathcal{C}$. In particular,* MIS *can be solved in FPT time if $\forall k' \leqslant k-1$ and $\forall H \subseteq_i G$, instances (H, k') can.*

Proof. Let S be an unknown but supposed independent set of G of size k. In time $h(k)n^c$ with $n := |V(G)|$, we compute a witness (A, B, R). For each $u \in R$, we branch on including u to our solution. This represents at most $g(k)$ branches with parameter $k - 1$. Now, we can focus on the case $S \cap R = \emptyset$.

We first deal separately with the special cases of $|S \cap A| = k$, $|S \cap B| = 0$ (a), and of $|S \cap A| = 0$, $|S \cap B| = k$ (b). As by assumption $|B| > kd(k)$, no maximal independent set has k vertices in A and zero in B. Indeed, by the one-sided almost disconnectedness, any k vertices in A dominate at most k^2 vertices in B. Hence at least one vertex of B could be added to this independent set of size k. So case (a) is actually impossible.

For case (b), we proceed as follows. We compute an independent set of size $k-1$ in $G[B]$. We temporary remove it from the graph, without removing its neighborhood. We compute a second independent set of size $k-1$ in $G[B]$ (without the first independent set); then a third one (in the graph deprived of the first two). We iterate this process until no independent set of size $k-1$ is found or we reach a total of $d(k)+1$ (disjoint) independent sets of size $k-1$ excavated in B. If we stop because of the former alternative, we know that an independent set of size k (actually even of size $k-1$) in B has to intersect the union of at most $d(k)$ independent sets of size $k-1$; so at most $(k-1)d(k)$ vertices in total. In that case, we branch on each vertex of this set of size at most $(k-1)d(k)$ with parameter $k-1$. If we stop because of the latter condition, we can include an arbitrary vertex w of A in the solution. By assumption, w has at least one neighbor in at most $d(k)$ independent sets of size $k-1$ in B. So at least one independent set of size $k-1$ of the collection is not adjacent to w, and forms with w a solution.

Now we are done with cases (a) and (b), we can assume that $k_1 := |S \cap A|$, $k_2 := |S \cap B| = k - k_1$ are both non-zero. Equivalently, $1 \leqslant k_1 \leqslant k-1$. We try out all the $k-1$ possibilities. For each, we perform a similar trick to the one used for case (b). We compute an independent set I_1 of size k_2 in $G[B]$. We then compute an independent set I_2 of size k_2 in $G[B \setminus I_1]$. Observe that there may be edges between I_1 and I_2. We compute an independent set I_3 in $G[B \setminus (I_1 \cup I_2)]$, and so on. We iterate this process until no independent set of size k_2 is found or we reach a total of $d(k)k_1+1$ (disjoint) independent sets of size k_2 excavated in B.

Say, we end up with the sets $I_1, \ldots, I_s$. Let $I := \bigcup_{j \in [s]} I_j$. If $s \leqslant f(k)k_1$, then we stopped because there was no independent set of size k_2 in $G[B \setminus I]$. This means that S intersects I. In that case, we branch on each vertex of I.

The other case is that $s = f(k)k_1 + 1$ and we stopped because we had enough sets I_j. In that case, we compute one independent set A_1 of size k_1 in $G[A]$. By assumption, $|N_B(A_1)| \leqslant k_1 d(k)$. In particular, there is at least one I_j which does not intersect $N_B(A_1)$. And $A_1 \cup I_j$ is our independent of size k.

Our algorithm makes at most

$$g(k) + d(k) + 1 + \sum_{k_1 \in [k-1]} (d(k)k_1 + 1) + 1 \leqslant g(k) + d(k) + 2 + k^2 d(k) + k$$

recursive calls to instances with parameter $k-1$, and we conclude by Lemma 5. ◀

Let $\mathcal{B}(A,B)$ be the bipartite graph between two disjoint vertex-subsets A and B (ignoring the edges internal to A and to B). We can further generalize the previous result to tripartitions (A,B,R) such that $\mathcal{B}(A,B)$ is $K_{d(k),d(k)}$-free.

▶ **Definition 19.** *Graphs in a class $\mathcal{C}$ are (g,d)-*weakly connected *if there exist two computable functions g and d, such that for every $G \in \mathcal{C}$ and $k \geqslant \alpha(G)$, there is a non-trivial tripartition (A,B,R) of $V(G)$ satisfying:*

- *$|R| \leqslant g(k)$,*
- *$|A|, |B| > \lceil d(k)^{d(k)} k^{2d(k)-1} \rceil + 1$, and*
- *$\mathcal{B}(A,B)$ is $K_{d(k),d(k)}$-free.*

Again, if we do not require $|A|$ and $|B|$ to be larger than $d(k)$, such a tripartition may trivially exist. We force A and B to be even larger than that to make the next theorem work. We show this theorem by combining ideas of the proof of Lemma 18 with the extremal theory result, known as Kővári-Sós-Turán's theorem, that $K_{t,t}$-free n-vertex graphs have at most $tn^{2-\frac{1}{t}}$ edges [24].

▶ **Theorem 20.** *Let $\mathcal{C}$ be a (g,d)-weakly connected class of graphs. Suppose for every $G \in \mathcal{C}$, a witness (A,B,R) of weakly connectedness can be found in time $h(k)|V(G)|^{\gamma}$. Then,* MIS *admits a decreasing FPT Turing-reduction in $\mathcal{C}$. In particular,* MIS *can be solved in FPT time if $\forall k' \leqslant k-1$ and $\forall H \subseteq_i G$, the instance (H,k') can.*

Proof. Let S be an unknown solution with $k_1 := S \cap A$ and $k_2 := S \cap B = k - k_1$. As previously, we try out all the $k+1$ values for k_1, setting k_2 to $k - k_1$. Let us first consider the $k-1$ branches in which $k_1 \neq 0$ and $k_2 \neq 0$.

Let $s := \lceil d(k)^{d(k)} k^{2d(k)-1} \rceil + 1$. Using the same process as in Lemma 18, we compute s disjoint independent sets $A_1, \ldots, A_s$ of size k_1 in $G[A]$ and s disjoint independent sets $B_1, \ldots, B_s$ of size k_2 in $G[B]$. Again, if the process stops before we reach s independent sets, we know that a solution (with k_1 vertices of A and k_2 vertices of B) intersects a set of size at most $k_1(s-1)$ or $k_2(s-1)$ and we can branch (since s is bounded by a function of k).

Now we claim that there is at least one pair (A_i, B_j) (among the s^2 pairs) without any edge between A_i and B_j; hence $A_i \cup B_j$ is an independent of size k. Suppose that this is not the case. Then, there is at least one edge between each pair (A_i, B_j). Therefore the bipartite graph $\mathcal{B} := \mathcal{B}(\bigcup_{i \in [s]} A_i, \bigcup_{i \in [s]} B_i)$ has at least s^2 edges, and $sk_1 + sk_2 = sk$ vertices. As $\mathcal{B}$ is also $K_{d(k),d(k)}$-free, it has, by Kővári-Sós-Turán's theorem, at most $d(k)(sk)^{2-\frac{1}{d(k)}}$ edges. But, by the choice of s, $s^2 > d(k)(sk)^{2-\frac{1}{d(k)}}$, a contradiction.

We now deal with the case $k_1 = 0$. We show that if a solution exists with $k_1 = 0, k_2 = k$, then the branch $k_1 = 1, k_2 = k-1$ also leads to a solution. Let us revisit that latter branch. We compute s disjoint independent sets $B_1, \ldots, B_s$ of size $k-1$ in $G[B]$. Again, if this process stops before we reach s independent sets, we can branch on each vertex of a set of size at most $(k-1)(s-1)$. This branching also covers the case $k_2 = k$, since clearly, an independent set of size k in $G[B]$ intersects those at most $(k-1)(s-1)$ vertices. Now, let A' be any set of s vertices in A and $\mathcal{B} := \mathcal{B}(A', \bigcup_{i \in [s]} B_i)$. By applying Kővári-Sós-Turán's theorem to $\mathcal{B}$ as in the previous paragraph, there should be at least one pair $(u, B_j) \in A' \times \{B_1, \ldots, B_s\}$ such that u is not adjacent to B_j.

We handle the case $k_2 = 0$ similarly, the conclusion being that we do not need to explore these branches. So we have described a decreasing FPT Turing-reduction creating less than $k(k+2)s$ instances (each with parameter $k' \leqslant k-1$), and we conclude by Lemma 5. ◀

A class of *co-graphs with parameterized noise* is a hereditary class in which all the graphs are almost bicomplete or almost disconnected. The following is a direct consequence of the previous lemmas.

▶ **Corollary 21.** *Given an FPT oracle finding the corresponding tripartitions,* MIS *is FPT in co-graphs with parameterized noise.*

The corollary still holds by replacing *almost disconnected* by *one-sided almost disconnected*, or even by *weakly connected*.

3.3 Summary and usage

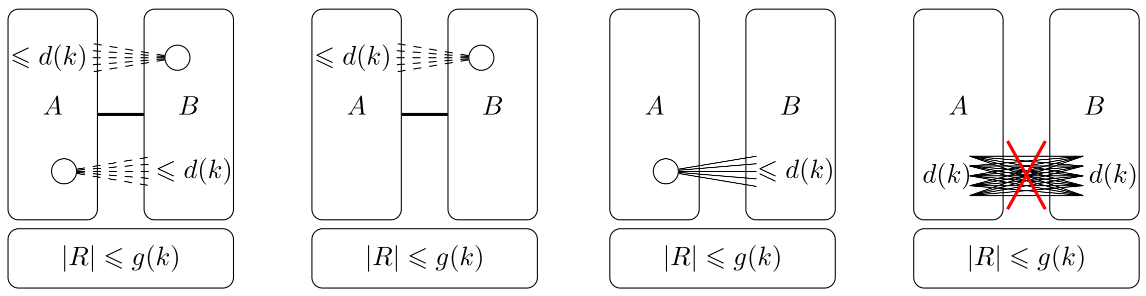
Figure 4 sums up the four FPT Turing-reductions that we obtained on almost disconnected and almost join graphs.

We know provide a few words in order to understand how to use these results. An obvious caveat is that, even if such a tripartition exists, computing it (or even, approximating it) may not be fixed-parameter tractable. What we hope is that on a class $\mathcal{C}$, we will manage to exploit the class structure in order to eventually find such tripartitions, in the cases we cannot conclude by more direct means. One of our main results, Theorem 22, illustrates that mechanism, when the algorithm is centered around getting to the hypotheses of Lemma 17 or Theorem 20.

4 FPT algorithm in $P(1, t, t, t)$-free graphs

We denote by $P(a, b, c, d)$ the graph made by substituting the vertices of P_4 by cliques of size a, b, c, and d, respectively. For instance, $P(1, 1, 1, 2)$ is $\overline{P}$ and $P(1, 1, 2, 1)$ is the kite. We settle the parameterized complexity of MIS on $\overline{P}$-free and kite-free graphs simultaneously (see Figure 3), by showing that MIS is FPT even in the much wider class of $P(1, t, t, t)$-free graphs.

▶ **Theorem 22.** *For every integer t,* MIS *is FPT in $P(1, t, t, t)$-free graphs.*



(a) Almost bicomplete tripartition, $A \neq \emptyset$, $B \neq \emptyset$.

(b) One-sided almost bicomplete, $S \cap A \neq \emptyset$ promise.

(c) One-sided almost disconnected, $|B| > kd(k), A \neq \emptyset$.

(d) Weakly connected, $\min(|A|, |B|) > (d(k)k^2)^{d(k)}$.

Figure 4 Summary of the FPT Turing-reductions and their hypotheses, provided we can efficiently find such tripartitions. For (c) and (d), the FPT Turing-reductions are decreasing, while for (a) and (b) they are just improving.

Proof. Let t be a fixed integer, and (G, k) be an input such that G is $P(1, t, t, t)$-free and $\alpha(G) \leqslant k$. We assume that $k \geqslant 3$, otherwise we conclude in polynomial-time.

The global strategy is the following. First we extract a collection C of disjoint and non-adjacent cliques with minimum and maximum size requirements, and some maximality condition. Then we partition the remaining vertices into equivalence classes with respect to their neighborhood in C. The maximum size imposed on the cliques of C ensures that the number of equivalence classes is bounded by a function of k. Setting C and the small[4] equivalence classes apart, we show that the rest of the graph is partitionable into (A, B) such that either $\mathcal{B}(A, B)$ is $K_{d(k),d(k)}$-free, in which case we conclude with Theorem 20, or $\mathcal{B}(A, B)$ is almost a complete bipartite graph, in which case we conclude with Lemma 17 (see Algorithm 2 for the pseudo-code).

As for the running time, we are looking for an algorithm in time $f(k)n^c$ for some fixed constant $c \geqslant 2$, and f an increasing computable function. We see f as a partial function on $[k-1]$, and extend it to $[k]$ in the recursive calls.

Building the clique collection C. For technical reasons, we want our collection C to contain at least two cliques, at least one of which being fairly large (larger than we can allow ourselves to brute-force). So we proceed in the following way. We find in polynomial-time $n^{8t+O(1)}$ a $2K_{4t}$. If G is $2K_{4t}$-free, an FPT algorithm already exists [4]. We see these two K_{4t} as the two initial cliques of our collection. Let X be the set of vertices with less than t neighbors in at least one of these two K_{4t}. We partition X into at most 2^{8t} vertex-sets (later they will be called *subclasses*) with the same neighborhood on the $2K_{4t}$. If all these sets contain less than $Ram(k+1, 2kt)$ vertices, X is fairly small: it contains less than $2^{8t}Ram(k+1, 2kt)$. The other vertices have at least t neighbors in both K_{4t}. We will show (Lemma 24) that this implies that these vertices are completely adjacent to both K_{4t}. Hence, vertices in the $2K_{4t}$ would have at most $2^{8t}Ram(k+1, 2kt)$ non-neighbors. In that case, we can safely remove the $2K_{4t}$ from G, by Observation 10.

So we can safely assume that (eventually) one subclass of X has more than $Ram(k+1, 2kt)$ vertices. We can find in polynomial-time a clique C_2 of size $2kt$. We build a new collection with $3t$ vertices of the first K_{4t}, that we name C_1. We take these vertices not adjacent to C_2 (this is possible since vertices in C_2 have the same at most $t-1$ neighbors in K_{4t}). Now we have in C a clique C_1 of size $3t$ and a clique C_2 of size $2kt$.

[4] the ones whose size is bounded by a later-specified function of k

We say that a clique of C is *large* if its size is above kt, and *small* otherwise. We can now specify the requirements on the collection C.

(1) C is a vertex-disjoint and independent[5] collection of cliques.
(2) all the cliques have size at least $3t$ and at most $2kt$.
(3) the number of cliques is at least 2.
(4) if we find a way to strictly increase the number of large cliques in C, we do it.

As $\alpha(G) \leqslant k$, the number of cliques in C cannot exceed k. This has two positive consequences. The first is in conjunction with the way we improve the collection C: by always increasing the number of large cliques by 1. Therefore, we can improve the collection C at most $k-1$ times. In particular, the improving process of C terminates (in polynomial time). The second benefit is that the total number of vertices of C is always bounded by $2k^2t$. Hence, the number of subclasses (sets of vertices with the exact same neighborhood in C) is bounded by a function of k (and the constant t).

As a slight abuse of notation, $C_1, \dots, C_s$ will always be the current collection C ($s < k$). We say that a vertex of $G - C$ *t-sees* a clique C_j of C if it has at least t neighbors in C_j. A *class* is a set of vertices t-seeing the same set of cliques of C. A *subclass* is a a set of vertices with the same neighborhood in C. Both classes and subclasses partition $G - C$. Observe that subclasses naturally refine classes. By extension, we say that a (sub)class t-sees a clique $C_i \in C$ if one vertex or equivalently all the vertices of that (sub)class t-see C_i.

Let $\eta := \lceil (2Ram(k+1,t))^{Ram(k+1,t)} 2^{2Ram(k+1,t)-1} \rceil + 1$. We choose this value so that $\eta^2/2 > Ram(k+1,t)(2\eta)^{2-1/Ram(k+1,t)}$ (it will become clear why in the proof of Lemma 27). We say that a subclass is *big* if it has more than $\max(Ram(k+1,2kt), \eta) = \eta$ vertices, and *small* otherwise. Since $\alpha(G) \leqslant k$, here are two convenient properties on a big subclass:

- a clique of size t can be found in polynomial-time, in order to build a potential $P(1,t,t,t)$,
- a clique of size $2kt$ can be found, in order to challenge the maximality of C.

We will come back to the significance of η later.

We can now specify item (4) of the clique-collection requirements. We resume where we left off the collection C, that is $\{C_1 = K_{3t}, C_2 = K_{2kt}\}$. While there is a big subclass that does not t-see any large clique of C, we find a clique of size $2kt$ in that subclass, and add it to the collection. We then remove the small clique (K_{3t}) potentially left, and in each large clique of C, we remove from C all neighbors of the subclass (they are at most $t-1$ many of them). This process adds a large clique to C, and decreases the size of the previous large cliques by at most $t-1$. Since the large cliques all enter C with size $2kt$, and the number of improvements is smaller than k, a large clique will remain large throughout the entire process. Therefore, the number of large cliques in C increases by 1. Since we started with one large clique among the first two cliques, the number of cliques remains at least 2. Note that, at each iteration, we update the subclasses with respect to the new collection C (see Algorithm 1 for the pseudo-code).

Small subclasses are set aside as their size is bounded by a function of k. Therefore, from hereon, all the subclasses are supposed big. We denote by $P(I)$ the class for which $I \subseteq [s]$ represents the indices of the cliques it t-sees. A first remark is that all the subclasses of $P(\emptyset)$ are small (so we "get rid of" the whole class $P(\emptyset)$).

▶ **Lemma 23.** *If P' is a subclass of $P(\emptyset)$, then $|P'| \leqslant Ram(k+1,2kt)$.*

Proof. P' does not t-see any (large) clique of C. So by the maximality property of C, it cannot contain a clique of size $2kt$ (see Algorithm 1). In particular, it cannot have more than $Ram(k+1,2kt)$ vertices. ◀

We turn our attention to classes $P(I)$ with $|I| \geqslant 1$ and their subclasses.

[5] there is no edge between two cliques of the collection

Algorithm 1 Routine for computing the clique collection C.

```
Precondition: $k$ is a positive integer, $G$ is not $2K_{4t}$-free, $\alpha(G) \leqslant k$
 1: function BuildCliqueCollection($G, k$):
 2:     $C \leftarrow \{K_{4t}, K_{4t}\}$                                  ▷ computed by brute-force
 3:     if $\exists$ big subclass not $t$-seeing both $K_{4t}$ then
 4:         $C_2 \leftarrow K_{2kt}$ in the subclass                       ▷ by Ramsey
 5:         $C_1 \leftarrow 3t$ vertices not adjacent to $C_2$ from one of the $K_{4t}$ not $t$-seen by the subclass
 6:         $C \leftarrow \{C_1, C_2\}$
 7:     else every big subclasses $t$-see both $K_{4t}$
 8:                 vertices in $C$ have bounded co-degree             ▷ Lemma 24
 9:                 we can safely delete them                          ▷ Observation 10
10:                 and call BuildCliqueCollection($G', k$) with the new graph $G'$
11:     end if
12:     while $\exists$ big subclass not $t$-seeing any large clique do
13:         $C_j \leftarrow K_{2kt}$ in the subclass                       ▷ by Ramsey
14:         $C' \leftarrow C \setminus \{\text{small clique}\}$            ▷ this is actually done at most once
15:         $C'' \leftarrow$ map($C'$,deleteNeighborsOf($C_j$))            ▷ remove $C_i \cap N(C_j)$ from each $C_i \in C$
16:         $C \leftarrow C'' \cup \{C_j\}$                                ▷ the new $C$ contains one more large clique, $C_j$.
17:     end while
18:     return $C$
19: end function
Postcondition: output $C$ is a collection of at least two (and at most $k-1$) pairwise independent cliques
    of size between $3t$ and $2tk$, and every big subclass $t$-sees at least one large clique (i.e., clique of $C$ of
    size at least $tk$).
```

Structure of the classes $P(I)$. We show a series of lemmas explaining how classes are connected to C and, more importantly, how they are connected to each other. This uses the ability to build cliques of size t at will, in big subclasses. Avoiding the formation of $P(1,t,t,t)$ will imply relatively dense or relatively sparse connections between classes $P(I)$ and $P(J)$.

▶ **Lemma 24.** *If a big subclass t-sees at least two cliques C_i and C_j of C, then all the vertices of that subclass are adjacent to all the vertices of both cliques.*

Proof. We find D, a clique of size t in the subclass. Let D_i and D_j be t neighbors of the subclass in C_i and C_j, respectively. Assume that the subclass has a non-neighbor $v \in C_i$. Then vD_iDD_j is a $P(1,t,t,t)$. ◀

In light of the previous lemma, if $|I| \geqslant 2$, the cliques of C that the class $P(I)$ t-sees are completely adjacent to $P(I)$.

▶ **Lemma 25.** *Let $I \subsetneq J \subseteq [s]$. Then, every vertex of $P(I)$ is adjacent to every vertex of $P(J)$ except at most $Ram(k+1,t)$.*

Proof. Let $i \in I$ and $j \in J \setminus I$. By Lemma 24, all vertices of $P(J)$ are adjacent to all vertices of $C_i \cup C_j$. Suppose, by contradiction, that there is a vertex $u \in P(I)$ with more than $Ram(k+1,t)$ non-neighbors in $P(J)$. We find a clique D of size t in $G[P(J) \setminus N(u)]$. Let D_i be t neighbors of u in C_i. Let $D_j \subset C_j$ be t neighbors of $P(J)$ which are not neighbors of u. Such a set D_j necessarily exists since u has at most $t-1$ neighbors in C_j, while $P(J)$ is completely adjacent to C_j, and $|C_j| \geqslant 3t$. Then uD_iDD_j is a $P(1,t,t,t)$. ◀

We say that two sets I, J are *incomparable* if I is not included in J, and J is not included in I. Recall that $\mathcal{B}(A, B)$ stands for the bipartite graph between vertex-set A and vertex-set B. Let $p(t, k) := 2^{2k^2 t}$ be a crude upper bound on the total number of subclasses.

▶ **Lemma 26.** *Let $I, J \subseteq [s]$ be two incomparable sets, and $P_\ell(I), P_{\ell'}(J)$ be any pair of subclasses of $P(I)$ and $P(J)$, respectively. Then, $\mathcal{B}(P_\ell(I), P_{\ell'}(J))$ is $K_{Ram(k+1,t),Ram(k+1,t)}$-free. Hence, $\mathcal{B}(P(I), P(J))$ is $K_{p(t,k)Ram(k+1,t),p(t,k)Ram(k+1,t)}$-free.*

Proof. Let $i \in I \setminus J$ and $j \in J \setminus I$. We first assume that one of I, J, say I, has at least two elements. Suppose, by contradiction, that there is a set $B_I \subseteq P_\ell(I)$ and a set $B_J \subseteq P_{\ell'}(J)$ both of size $Ram(k+1, t)$, such that there is no non-edge between B_I and B_J. Let u be a vertex of C_j which is adjacent to $P_{\ell'}(J)$ but not to $P_\ell(I)$. We find D_I, a clique of size t in $G[B_I]$, and D_J, a clique of size t in $G[B_J]$. Let D_i be t neighbors of $P_\ell(I)$ in C_i that are not adjacent to $P_{\ell'}(J)$. Those t vertices exist since, by Lemma 24, $P_\ell(I)$ is completely adjacent to C_i (by assumption $|I| \geqslant 2$). And $P_{\ell'}(J)$ has more than t non-neighbors in C_i. Then, $uD_JD_ID_i$ is a $P(1, t, t, t)$.

We now have to settle the remaining case: $|I| = |J| = 1$ ($I = \{i\}$ and $J = \{j\}$). If $P_\ell(I)$ has at least $2t$ neighbors in C_i or $P_{\ell'}(J)$ has at least $2t$ neighbors in C_j, we conclude as in the previous paragraph. So we assume that it is not the case. We distinguish two cases.

Either $P_\ell(I)$ has at least one neighbor in C_j, say u. Let D_I be a clique of size t in $P_\ell(I)$, $D_i \subseteq C_i$ be t neighbors of $P_\ell(I)$, and $D'_i \subseteq C_i$ be t non-neighbors of $P_\ell(I)$. D_i and D'_i exist since $P_\ell(I)$ has between t and $2t-1$ neighbors in C_j, and $|C_j| \geqslant 3t$. Then, $uD_ID_iD'_i$ is a $P(1, t, t, t)$.

Or $P_\ell(I)$ has no neighbor in C_j. Let u be a non-neighbor of $P_{\ell'}(J)$ in C_j, and $D_j \subseteq C_j$ be t neighbors of $P_{\ell'}(J)$. If there is a set $B_I \subseteq P_\ell(I)$ and a set $B_J \subseteq P_{\ell'}(J)$ both of size $Ram(k+1, t)$, such that B_I and B_J are completely adjacent to each other. We can find D_I, a clique of size t in $G[B_I]$, and D_J, a clique of size t in $G[B_J]$. Then, $uD_jD_JD_I$ is a $P(1, t, t, t)$. This implies that, in any case, there cannot be a $K_{p(t,k)Ram(k+1,t),p(t,k)Ram(k+1,t)}$ in $\mathcal{B}(P(I), P(J))$. ◀

We say that the sets I and J *overlap* if all three of $I \cap J$, $I \setminus J$, $J \setminus I$ are non-empty.

▶ **Lemma 27.** *Let $I, J \subseteq [s]$ be two overlapping sets. Then, at least one of $P(I)$ and $P(J)$ have only small subclasses.*

Proof. Suppose, by contradiction, that there is a big subclass $P_\ell(I)$ of $P(I)$, and a big subclass $P_{\ell'}(J)$ of $P(J)$. Observe that, for I and J to overlap, their size should be at least 2. Let $i \in I \setminus J$, $j \in J \setminus I$, $h \in I \cap J$. By the arguments of Lemma 25 applied to the restriction to $P(I)$, $P(J)$, C_h, and C_j, a vertex in $P(I)$ has at most $Ram(k+1, t)$ non-neighbors in $P(J)$. Let us consider η vertices in $P_\ell(I)$ and η vertices in $P_{\ell'}(J)$. Since $\eta \geqslant 2Ram(k+1, t)$, the previous observation implies that the number of edges between them is at least $\eta^2/2$. But by Lemma 26, the bipartite graph linking them should be $K_{Ram(k+1,t),Ram(k+1,t)}$-free. By Kővári-Sós-Turán's theorem, this number of edges is bounded from above by $Ram(k+1, t)(2\eta)^{2-1/Ram(k+1,t)} < \eta^2/2$, a contradiction. ◀

Hence, the remaining (not entirely made of small subclasses) classes define a laminar[6] set-system. We denote by R the union of the vertices in all the small subclasses and C. We now add a new condition to be a small subclass (condition that we did not need thus far). A subclass is also small if it has at most $|R|$ vertices. Note that this condition can snowball. But eventually R has size bounded by $g(k) := 2^{p(t,k)}(p(t, k)\eta + 2k^2 t)$. A class is *remaining* if it contains at least one big subclass. By Lemma 23, $P(\emptyset)$ cannot be remaining. If no class is remaining, then the whole graph is a kernel. So we can assume that there is at least one remaining class. Let $P(I)$ be a remaining class in $G - R$ such that I is maximal

[6] where two sets are nested or disjoint

among the remaining classes. We distinguish two cases: either there is at least one other remaining class $P(J)$ $(I \neq J)$, or $P(I)$ is the unique remaining class.

At least two remaining classes $P(I)$ and $P(J)$. By Lemma 27, any other class $P(J)$ satisfies $J \subsetneq I$ or $I \cap J = \emptyset$. Let $\iota, \delta \leqslant 2^k$ be the number of remaining classes such that $J \subsetneq I$ and such that $I \cap J = \emptyset$, respectively. Again, we distinguish two cases: $\delta > 0$, and $\delta = 0$. If $\delta > 0$, we build the partition (A, B, R) of $V(G)$ such that A contains the $\iota+1$ classes whose set is included in I and B contains the δ classes whose set is disjoint from I. By Lemma 26, the bipartite graph between any of the $(\iota+1)\delta$ pairs of classes made of one class whose set is contained in I and one class whose set is disjoint from I is $K_{p(t,k)Ram(k+1,t),p(t,k)Ram(k+1,t)}$-free. Hence, the bipartite graph between A and B is $K_{2^k p(t,k)Ram(k+1,t),2^k p(t,k)Ram(k+1,t)}$-free. Thus we conclude by Theorem 20 with $d(k) = 2^k p(t,k)Ram(k+1,t)$.

We now tackle the case $\delta = 0$, that is, all the remaining classes $P(J)$ satisfy $J \subseteq I$. We first assume that there are two remaining classes with disjoint sets. A laminar set-system with a unique maximal set can be represented as a rooted tree, where nodes are in one-to-one correspondence with the sets, and the parent-to-child arrow represents the partial order of inclusion. Here, the root is labeled by I (since I is the unique maximal set), and all the nodes are labeled by a subset of $[s]$ corresponding to a remaining class. Let $I = I_1 \supsetneq I_2 \supsetneq \ldots \supsetneq I_h$ be the path from the root to the first node with out-degree at least 2. Observe that C contains at most k cliques, so $h \leqslant k$. Let $J_1, J_2, \ldots, J_\ell$ be the ℓ children of I_h (with $\ell \geqslant 2$). Let $\mathcal{P}_1$ be the remaining classes whose set is included in J_1, and $\mathcal{P}_{2+}$ be the remaining classes whose set is included in one J_i for some $i \in [2, \ell]$. Let $A := \bigcup_{q \in [h]} P(I_q)$, and $B := V(G) \setminus (A \cup R)$. By Lemma 25, vertices of B have at most $hRam(k+1,t) \leqslant kRam(k+1,t)$ non-neighbors in A. We apply Lemma 17 with the tripartition (A, B, R) and $d_1(k) = kRam(k+1,t)$. Only we did not cover the case in which the solution does not intersect A. We do so by applying Theorem 20 to the tripartition $(\mathcal{P}_1, \mathcal{P}_{2+}, R)$ with $d_2(k) = 2^k p(t,k)Ram(k+1,t)$. A priori, what we just did is not bounded by $f(k)|V(G)|^c$, hence not legal. Let us go back to the last lines of Lemma 17 and of Theorem 20. Our running time is bounded by $f(k)|A \cup R|^c + k^2\binom{d_1(k)}{k}d_1(k)^c f(k-1)|B|^c + k(k+2)(\lceil d_2(k)^{d_2(k)}k^{2d_2(k)-1}\rceil + 1)f(k-1)|B|^c$, where the two first terms come from the application of Lemma 17, and the third term, from Theorem 20. This is at most $f(k)|A \cup R|^c + f(k)|B|^c \leqslant f(k)|V(G)|^c$ by Cauchy-Schwarz inequality, with $f(k) := (k^2\binom{d_1(k)}{k}d_1(k)^c + k(k+2)(\lceil d_2(k)^{d_2(k)}k^{2d_2(k)-1}\rceil + 1))f(k-1)$.

Let now assume that all the remaining classes have nested sets (no two sets are disjoint). Let $I = I_1 \supsetneq I_2 \supsetneq \ldots \supsetneq I_h$ the sets of *all* the remaining classes $(h \leqslant k)$. Suppose $h \geqslant 3$. We apply Lemma 17 to the tripartition $(P(I_1) \cup P(I_2), \bigcup_{j \in [3,h]} P(I_j), R)$ with $d(k) = 2Ram(k+1,t)$. Indeed, by Lemma 25, vertices of $\bigcup_{j \in [3,h]} P(I_j)$ have at most $Ram(k+1,t)$ non-neighbors in $P(I_1)$ and at most $Ram(k+1,t)$ non-neighbors in $P(I_2)$. We deal with the case in which the solution does not intersect $P(I_1) \cup P(I_2)$ in the following way. Let C_q be the clique of C only t-seen by $P(I_1)$ and $C_{q'}$ the clique of C only t-seen by $P(I_1) \cup P(I_2)$. One of these two cliques has to be large (since there is at most one small clique). We branch on the at least tk and at most $2tk$ vertices of that large clique, say C'. A maximal independent set cannot be fully contained in $\bigcup_{j \in [3,h]} P(I_j)$. Indeed, any choice of at most k vertices in this set dominates at most $k(t-1)$ vertices of C'. Thus, we cannot miss a solution. Let us turn to the running time. Once again, we cannot use Lemma 17 as a total black-box. Our running time is bounded by $f(k)|A \cup R|^c + k^2\binom{d(k)}{k}d(k)^c f(k-1)|B|^c + 2tkf(k-1)|B \cup R|^c \leqslant f(k)|A \cup R|^c + f(k)|B \cup R|^c$ with $f(k) := (k^2\binom{d(k)}{k}d(k)^c + 2tkf)f(k-1)$, and $f(k)|A \cup R|^c + f(k)|B \cup R|^c \leqslant f(k)|V(G)|^c$, by Lemma 15. Here we need that $|A| > |R|$ and $|B| > |R|$ which is the case: recall that we added that requirement to be a big subclass.

The last case is the following. There are exactly two remaining classes associated to sets $I = I_1 \supsetneq I_2$. If a clique not t-seen by $P(I_2)$ is large or if $P(I_2)$ is $2K_{4t}$-free, we conclude with Lemma 17 (recall that this finds a solution if there is one intersecting $P(I_1)$. In both cases, if the solution does not intersect $P(I_1)$, we can find it with only a small overhead cost. If a clique not t-seen by $P(I_2)$ is large, we branch on the at most $2kt$ vertices of that clique. If $P(I_2)$ is $2K_{4t}$-free, an independent set of size k can be found in $G[P(I_2)]$ in FPT time [4].

Finally, we can assume that $G[P(I_2)]$ contains a $2K_{4t,4t}$ and does not t-see a small clique in C. Note that this implies that C is made of two cliques K_{3t} and K_{2kt}. We call *critical* such a case where $C = \{K_{3t}, K_{2kt}\}$ and a $2K_{4t}$ can be found in a class not t-seeing K_{3t}.

For this very specific case (that may also arise with a unique remaining class, see below), we perform the following refinement of the clique-collection computation. We compute a new clique collection, say C^2, in $G - C$, starting with a $2K_{4t,4t}$ found in the class not t-seeing the previous K_{3t}. If C^2 is not of the form $\{K_{3t}, K_{2kt}\}$, we add C to the bounded-in-k set R, and we follow our algorithm (that is, a non-critical case). If $C^2 = \{K_{3t}, K_{2kt}\}$, we compute a new clique collection C^3 in $G - (C^1 \cup C^2)$ (with $C^1 = C$), again starting with a $2K_{4t,4t}$ found in the class not t-seeing the previous K_{3t}, and so on. Let us assume that we are always in a critical case, with $C^h = \{C_1^h = K_{3t}, C_2^h = K_{2kt}\}$. We stop after $\zeta := Ram_{2^{(3t)^2}}(4kt)$ iterations, leading to disjoint (though not independent) clique collections $C = C^1, C^2, \ldots, C^\zeta$. In particular, $|\bigcup_{h \in \zeta} C^h|$ is still bounded by a function of k, namely $\zeta(3t + 2kt)$. We claim that we can find a $2K_{2kt,2kt}$ in $G[\bigcup_{h\in\zeta} C_1^h]$.

Because of the number of iterations, one can extract $4kt$ cliques C_1^h (of size $3t$) with the same bipartite graph linking any pair of C_1^h (with a fixed but arbitrary ordering of each C_1^h). This common bipartite graph has to be empty, complete, or a half-graph. Let us show that it can only be a half-graph. For any $i \in [3t]$, the i-th vertices in the C_1^h should be adjacent (otherwise they form an independent set of size $2kt$). That excludes the empty bipartite graph. Let h_1 be the smallest index such that we have extracted $C_1^{h_1}$. The common bipartite graph cannot be complete either, since all the vertices of $G - (\bigcup_{h\in[h_1]})$ have at most $t-1$ neighbors in $C_1^{h_1}$. This was one of the condition of a critical case. So the bipartite graph is a half-graph. Then we find our $2K_{2kt,2kt}$ as the first vertex (or last vertex) of the first $2kt$ extracted cliques, and the last vertex (or first vertex) of the last $2kt$ extracted cliques. Now we finally have a clique collection with two independent *large* cliques, depending on the orientation of the half-graph. So we can start again without reaching the problematic case.

Unique remaining $P(I)$. If $|I| \geqslant 2$, by Lemma 24, $P(I)$ is completely adjacent to one clique C_i (with $i \in I$). Any vertex of C_i has at most $g(k)$ non-neighbors. This case is handled by Observation 10. So we now suppose that $|I| = 1$ (and $I = \{i\}$). If $P(I)$ does not t-see a large clique C_j, we can branch on the at most $2kt$ vertices of that clique. Indeed, there is a solution that intersects it, since $k-1$ vertices in $G - R$ can dominate at most $(k-1)(t-1) < kt$ vertices. Thus, we can further assume that $P(I)$ t-sees all the large cliques. This forces that there is at most one large clique, since $|I| = 1$. There cannot be at least three cliques in C. Indeed, the way the collection is maintained, that would imply that there are at least two large cliques. So, $C = \{C_1 = K_{3t}, C_2 = K_{2kt}\}$ and $I = \{2\}$. This is a *critical* case, which we handle as in the previous paragraph (with two remaining classes). ◀

5 Randomized FPT algorithms in dart-free and cricket-free graphs

In this section, we consider the case of dart-free and cricket-free graphs, and prove that there is a randomized FPT algorithm for MIS in both graph classes. To this end, we use the technique of iterative expansion together with a Ramsey extraction.

Let us first define the Ramsey extraction. Notice that for all results presented in this paper, the cliques will be of size 1 or 2 only, which greatly simplifies the construction. However, we give the complete version of the definition, as introduced in [4].

▶ **Definition 28.** *Given a graph G and a set of $k-1$ vertex-disjoint cliques of G, $\mathcal{C} = \{C_1, \ldots, C_{k-1}\}$, each of size q, we say that $\mathcal{C}$ is a set of* Ramsey-extracted cliques of size q *if the conditions below hold. Let $C_r = \{c_j^r : j \in \{1, \ldots, q\}\}$ for every $r \in \{1, \ldots, k-1\}$.*

Algorithm 2 FPT algorithm for MIS on $P(1,t,t,t)$-free graphs

```
Precondition: $G$ is $P(1,t,t,t)$-free, $k \geqslant \alpha(G)$
1: function STABLE($G, k$):
2:     if $k \leqslant 2$ then solve in $n^2$ by brute-force
3:     end if                                                        ▷ now $k \geqslant 3$
4:     if $G$ is $2K_{4t}$-free then solve in FPT time
5:     end if                                                        ▷ see [4]
6:     $C \leftarrow$ BuildCliqueCollection$(G, k)$
7:     $R \leftarrow C \cup$ subclasses of size less than $\eta$     ▷ small subclasses are set aside
8:     while $\exists$ subclass $Q$ of size at most $|R|$ do
9:         $R \leftarrow R \cup Q$
10:    end while
11:    $\mathcal{P} \leftarrow$ remaining classes
12:    if $\mathcal{P} = \emptyset$ then input is a kernel
13:    end if
14:    $P(I) \leftarrow$ remaining class with $I$ maximal for inclusion
15:    if $|\mathcal{P}| \geqslant 2$ then
16:        if $\exists P(J) \in \mathcal{P}$ such that $I \cap J = \emptyset$ then
17:            $(A, B, R)$ with $\mathcal{B}(A, B)$ $K_{d(k),d(k)}$-free        ▷ Theorem 20
18:        end if
19:        if $\forall P(J) \in \mathcal{P}$, $J \subseteq I$ then
20:            $(A, B, R)$ with $\forall v \in B$, $v$ has co-degree $\leqslant d_1(k)$ in $A$     ▷ Lemma 17
21:            and $(B_1, B_2, R)$ in $G[B \cup R]$ with $\mathcal{B}(B_1, B_2)$ $K_{d_2(k),d_2(k)}$-free,   ▷ Theorem 20
22:            or branching on $2tk$ vertices,
23:            or critical case, when repeated, yields a $2K_{2kt,2kt}$
24:        end if
25:    end if
26:    if $\mathcal{P} = \{P(I)\}$ then a vertex of $C$ has small co-degree,     ▷ see Observation 10
27:                                  or branching on $2tk$ vertices,
28:                                  or critical case, when repeated, yields a $2K_{2kt,2kt}$
29:    end if
30: end function
```

- *For every $j \in [q]$, the set $\{c_j^r : r \in \{1, \ldots, k-1\}\}$ is an independent set of G of size $k-1$.*
- *For any $r \neq r' \in \{1, \ldots, k-1\}$, one of the four following case can happen:*

  (i) *for every $j, j' \in [q]$, $c_j^r c_{j'}^{r'} \notin E(G)$*

  (ii) *for every $j, j' \in [q]$, $c_j^r c_{j'}^{r'} \in E(G)$ iff $j \neq j'$*

  (iii) *for every $j, j' \in [q]$, $c_j^r c_{j'}^{r'} \in E(G)$ iff $j < j'$*

  (iv) *for every $j, j' \in [q]$, $c_j^r c_{j'}^{r'} \in E(G)$ iff $j > j'$*

  In the case (i) (resp. (ii)), we say that the relation between C_r and $C_{r'}$ is empty *(resp.* full[7]*). In case (iii) or (iv), we say the relation is* semi-full.

We then define the following problem:

[7] Remark that in this case, the graph induced by $C_r \cup C_{r'}$ is the complement of a perfect matching.

▶ **Definition 29.** *The f-*Ramsey-extracted Iterative Expansion MIS *problem takes as input an integer k and a graph G whose vertices are partitioned into non-empty sets $X_1 \cup \dots \cup X_k \cup C_1 \cup \dots \cup C_{k-1}$, where:*

- *$\{C_1, \dots, C_{k-1}\}$ is a set of $k-1$ Ramsey-extracted cliques of size $f(k)$*
- *any independent set of size k in G is contained in $X_1 \cup \dots \cup X_k$*
- *$\forall i \in \{1, \dots, k\}$, $\forall v, w \in X_i$ and $\forall j \in \{1, \dots, k-1\}$, $N(v) \cap C_j = N(w) \cap C_j = \emptyset$ or $N(v) \cap C_j = N(w) \cap C_j = C_j$*
- *the following bipartite graph $\mathcal{B}$ is connected: $V(\mathcal{B}) = B_1 \cup B_2$, $B_1 = \{b^1_1, \dots, b^1_k\}$, $B_2 = \{b^2_1, \dots, b^2_{k-1}\}$ and $b^1_j b^2_r \in E(\mathcal{B})$ iff X_j and C_r are adjacent.*

The objective is the following:

- *if G contains an independent set S such that $S \cap X_i \neq \emptyset$ for all $i \in \{1, \dots, k\}$, then the algorithm must answer "YES". In that case the solution is called a* rainbow independent set*.*
- *if G does not contain an independent set of size k, then the algorithm must answer "NO".*

As well as the following result from [4], which allows to focus on f-Ramsey-extracted Iterative Expansion MIS.

▶ **Theorem 30.** *[4] Let $\mathcal{G}$ be a hereditary graph class. If f-*Ramsey-extracted Iterative Expansion MIS *is FPT in $\mathcal{G}$ for some computable function f, then* MIS *is FPT in $\mathcal{G}$.*

Hence, in the following, by *positive instance* we mean an instance having a rainbow independent set, while *negative instance* denotes instances not containing an independent set of size k. If the input graph contains at least one independent set of size k, but none of them is rainbow, then we are allowed to output "No" (this case is handled by the color coding technique in the proof of the previous result).

For both dart-free and cricket-free graphs, our algorithms will consist of a bounded search tree solving f-Ramsey-extracted Iterative Expansion MIS. Let us first describe some common ingredients of both algorithms. Throughout the proof, $I = (G, k)$ denote our input of f-Ramsey-extracted Iterative Expansion MIS.

The *set-graph* R_G is the graph on k vertices obtained by contracting all X_i's to single vertices, putting an edge between X_i and X_j if there exists an edge between these two sets.

Our proof consists of a bounded search tree. To this end, we will rely on branchings, which will decrease the parameter $\kappa(G, k)$ defined in the following lexicographic order:

1. k
2. $|E(R_G)|$
3. $\sum_{i=1}^{k} \alpha(G[X_i])$

Notice that the number of possible values of $\kappa(G, k)$ is bounded by a function of k only. The idea of our algorithm is to restrict more and more the structure of an instance by the mean of branchings. To this end, our branchings will either strictly decrease our parameter $\kappa(I)$, or create instances with particular properties. This is formalized by the following definition.

▶ **Definition 31.** *Given an instance I, we say that we can FPT-reduce (or simply* reduce*) to some set of instances $\mathcal{I}$ if there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that we can output a set of instances $I_1, \dots, I_{g(k)}$ such that for every $p \in [g(k)]$, either $\kappa(I_p) < \kappa(I)$, or $I_p \in \mathcal{I}$. This step must run in time polynomial in $\sum_{p=1}^{g(k)} |I_p|$.*

At some places, we will sometimes make use of *random reductions*: the instance I will be transformed into another instance I' with a random rule. We will then ensure that $\kappa(I') < \kappa(I)$, and:

- if I is a positive instance (the graph admits a rainbow independent set), then I' is a positive instance with probability at least $h(k)$ for some computable function h
- if I is a negative instance, then I' is a negative instance.

It is easy to see that since the parameter κ decreases, the resulting algorithm is a one-sided error Monte Carlo algorithm with a success probability depending on k only. Hence, both algorithms will be *randomized FPT algorithms.*

We begin with a branching whose purpose is to "clean" some carefully chosen adjacencies of the set graph.

▶ **Definition 32.** *We say that a couple $(i,j) \in [k]^2$ is* clean *if the following conditions are satisfied:*

- *for every $x \in X_i$, x has a non-neighbor in X_j*
- *for every $x \in X_i$, x has a neighbor in X_j*
- *$G[X_i]$ is connected.*

Observe that if a couple (i,j) is clean, then there exists xy in $E(G[X_i])$ and $z \in X_j$ such that $xz \in E(G)$ but $yz \notin E(G)$. We say that a set of couples $\mathcal{P} \subseteq [k]^2$ is *acyclic* if the oriented graph whose vertices are $[k]$ and arcs are $\mathcal{P}$ is acyclic.

▶ **Lemma 33.** *Given an acyclic set of couples $\mathcal{P} \subseteq [k]^2$ corresponding to some edges of the set graph, one can FPT-reduce to instances where every couple $(i,j) \in \mathcal{P}$ is clean.*

Proof. Let $\mathcal{P} = \{p_1, \ldots, p_t\}$ considered in a total ordering (recall that $\mathcal{P}$ is acyclic). Iteratively for ℓ from t downto 1, we FPT-reduce to an instance where couples p_ℓ, ..., p_t are clean. Let $p_\ell = (i,j)$. If X_i and X_j are already clean, we are done. Otherwise, let $U \subseteq X_i$ be the vertices x of X_i such that $xz \in E(G)$ for every $z \in X_j$, let $S \subseteq X_i$ be the vertices x of X_i such that $xz \notin E(G)$ for every $z \in X_j$, and let Ω_1, ..., Ω_q be the connected components of $G[X_i \setminus (U \cup S)]$ (we may assume $q < k$, otherwise there is an independent set of size k in G). We output the following new instances:

- I_S, where X_i is replaced by S
- I_{Ω_r}, where X_i is replaced by Ω_r, for every $r \in [q]$.

In every new instance, the graph is an induced subgraph of the former one, hence, if I is negative, all new instances are negative as well. Moreover, if there is a rainbow independent set in I, it must intersect either S or Ω_r for some $r \in [q]$ (it cannot intersect U), hence if I is positive then one of the new instances is positive. Now, observe that $\kappa(I_S) < \kappa(I)$ since there is no edge between S and X_j (while there was an edge between X_i and X_j). Moreover, if $q > 1$, then $\kappa(I_{\Omega_r}) < \kappa(I)$ for every $r \in [q]$, since the size of a maximum independent set in $G[\omega_r]$ has decreased. Finally, if $q = 1$, in I_{Ω_1}, the couples p_ℓ, ..., p_t are clean, since we only modified X_i, and i does not appear in p_s for every $s > \ell$ (recall that we consider couples of $\mathcal{P}$ in an inverted total ordering). ◀

One simple case is where the set graph is a cycle or a path:

▶ **Lemma 34** (Particular set graph: cycle or path)**.** *If the set graph is a path or a cycle, then f-*Ramsey-extracted Iterative Expansion MIS *is polynomial-time solvable for every computable function f.*

Proof. Assume the set graph is a path, with edges between X_i and X_{i+1}, $i \in [k-1]$. In that case, observe that there is a solution if and only if the following dynamic programming returns *true* on input $P(2, x_1)$ for some $x_1 \in X_1$:

$$P(i, x_{i-1}) = \begin{cases} true & \text{if } i = k \\ false & \text{if } X_i \subseteq N(x_{i-1}) \\ \bigvee_{x_i \in X_i \setminus N(x_{i-1})} P(i+1, x_i) & \text{otherwise.} \end{cases}$$

Clearly this dynamic programming runs in $O(mnk)$ time, where m and n are the number of edges and vertices of the graph induced by $\cup_{i=1}^{k} X_i$, respectively. Similar ideas can be used when the set graph is a cycle. ◀

We are now ready to present our algorithms for dart-free and cricket-free graphs.

5.1 The dart

▶ **Theorem 35.** *There is a randomized FPT algorithm for* MIS *in dart-free graphs parameterized by the size of the solution.*

Proof. As said previously, we solve f-Ramsey-extracted Iterative Expansion MIS in dart-free graphs. Here, we define $f(x) = 1$ for every $x \in \mathbb{N}$. Hence, for every $j \in [k-1]$, $C_j = \{c_j\}$. The strategy is to use FPT branching and random reductions in order to simplify the structure of the set graph. More precisely, the goal is to reduce to the case where the set graph is {paw, claw}-free[8].

▶ **Lemma 36** (Removal of some P_3's: part I)**.** *One can FPT-reduce to instances where for every triple (X_1, X_2, X_3) of the set graph inducing a P_3, no vertex of $\{c_1, \ldots, c_{k-1}\}$ is adjacent to X_1, X_2 and X_3.*

Proof. Suppose that this is the case, and let c_j be such a vertex. Then apply the branching of Lemma 33 with couples $\{(1,2),(2,3)\}$. We end up with an instance where there exists an edge xy induced by X_1 and a vertex $z \in X_2$ such that $xz \in E(G)$ but $yz \notin E(G)$, and a vertex $u \in X_3$ such that $zu \notin E(G)$. But observe that $\{c_j, x, y, z, u\}$ induces a dart, which is impossible. ◀

▶ **Lemma 37** (Removal of some P_3's: part II)**.** *One can FPT-reduce to instances where for every triple (X_1, X_2, X_3) of the set graph inducing a P_3, no vertex of $\{c_1, \ldots, c_{k-1}\}$ is adjacent to X_2 but not to X_1 nor X_3.*

Proof. Suppose that this is the case: let c_j be such a vertex. Apply Lemma 33 with couples $\{(2,1),(2,3)\}$. We end up with an instance where every vertex $x \in X_2$ has a neighbor in X_1 and X_3. But observe that for every edge xy induced by X_2, x and y must be a module with respect to X_1 and X_3. Indeed, if there is a vertex $u \in X_1$ both adjacent to x and y, then for every neighbor $z \in X_3$ of x, z must also be a neighbor of y, for otherwise $\{c_j, x, y, u, z\}$ would induce a dart. But then it means that $z \in X_3$ is adjacent to both x and y, and for the same reasons, the neighborhood of x and y in X_1 must be the same. Hence, we can partition X_2 into subsets $\{M_i\}_{i=1}^{p}$, each of which being both a maximal module with respect to X_1 and X_3. For every $i \in [p]$, denote by N_i the set of common neighbors of M_i in X_1. Note that the previous paragraph ensures that if there is an edge between M_i and M_j, then N_i and N_j must be disjoint. Now let us consider any vertex $x \in X_1$. If x is adjacent to both a vertex of M_i and M_j (with $i \neq j$), then, since x is complete to these sets, it means that there is no edge between M_i and M_j since $N_i \cap N_j \neq \emptyset$. In particular, x is only adjacent to at most $k-1$ modules M_i (since otherwise we would be able to find in polynomial time an independent set of size k in X_2).

We now proceed to a random reduction: for every $i \in [p]$, we delete all vertices of M_i with probability $1/2$, and after that, we remove every vertex $x \in X_1$ if it has a remaining neighbor in X_2. Assume the instance is positive: let $s_1 \in X_1$ and $s_2 \in X_2$ be the elements of a rainbow independent set. Observe that the probability that s_2 has not been removed is at least $1/2$, while the probability that s_1 has not been removed is at least $1/2^{k-1}$ (since s_1 is adjacent to at most $k-1$ modules, and s_1 is kept if all its possible neighbors have been deleted). Hence, after this removal step, the obtained instance is positive with probability at least $1/2^k$ (and if the instance was negative, the reduced one shall be negative as well, since the reduced graph is an induced subgraph of the former). Moreover, in the reduced instance there is no edge between X_1 and X_2, hence the parameter κ decreases. ◀

▶ **Lemma 38** (No claw in the set graph)**.** *One can FPT-reduce to instances where the set graph is claw-free.*

[8] Recall that the paw is the graph obtained by adding one edge to the claw.

Proof. We apply Lemmas 36 and 37, and claim that we end up with instances where the set graph is claw-free. Suppose it is not: let X_1 be the center of the claw, and let $c_j \in \{c_1, \dots, c_{k-1}\}$ adjacent to X_1. There are two cases: c_j is also adjacent to at least two neighbors of X_1, but this is impossible by Lemma 36. The other case is also impossible by Lemma 37. ◀

▶ **Lemma 39** (Removal of paws in the set graph)**.** *One can FPT-reduce to instances where the set graph is paw-free.*

Proof. Let X_1, X_2, X_3, X_4 be a paw of the set graph such that the non-edges are between X_4 and $X_2 \cup X_3$. Let $c_j \in \{c_1, \dots, c_{k-1}\}$ be a neighbor of X_1 (recall that each X_i is a module *w.r.t.* $\{c_1, \dots, c_{k-1}\}$). We first apply Lemmas 36 and 37, and Lemma 33 for the couples $\{(1,4),(1,2)\}$. Hereafter, there are only two cases: either c_j is also adjacent to X_2, X_3 but not to X_4, or is adjacent to X_4 but not to X_2, X_3.

Case 1: c_j is adjacent to X_2, X_3 but not to X_4. Let u be an arbitrary vertex of X_1, and assume the instance is positive: let $s_a \in X_a$ be the elements of a rainbow independent set for $a \in \{1,2,3\}$. We have the following:

- u cannot be adjacent to both s_2 and s_3 since, in this case, $\{c_j, s_2, s_3, u\}$ together with a neighbor of u in X_4 (remember that we applied Lemma 33 for the couple $(1,4)$, hence such a neighbor must exist) induce a dart.
- u cannot be adjacent to s_1 and to exactly one vertex among $\{s_2, s_3\}$, since, in this case, $\{u, s_1, s_2, s_3, c_j\}$ induces a dart.

We now create four branches which will correspond to the remaining different possibilities of adjacencies between u and $\{s_1, s_2, s_3\}$: the first three branches (first item below) represent the case where u is adjacent to only one vertex among $\{s_1, s_2, s_3\}$, while the second item represents the case where u is adjacent to none of $\{s_1, s_2, s_3\}$:

- for every $a \in \{1,2,3\}$, create a branch where:
  - X_a is replaced by $X_a \cap N[u]$
  - X_b is replaced by $X_b \setminus N(u)$, for $b \neq a$.

  And apply Lemma 33 with a couple (b,c) with $b, c \neq a$ (chosen arbitrarily).
- in the fourth branch, replace X_a by $X_a \setminus N(u)$ for every $a \in \{1,2,3\}$, and apply Lemma 33 with the couple $(2,3)$.

Now, observe that:

- if u is adjacent to only s_a, then $\{s_1, s_2, s_3\}$ are still in the reduced graph of the corresponding first item above, and, moreover, there is no edge between X_b and X_c, since, because of Lemma 33, if there was an edge between X_b and X_c, we would be able to find an edge xy in X_b and a vertex z in X_v such that xz is an edge but yz is not, and $\{x, y, z, u, c_j\}$ would induce a dart (recall that u is not adjacent to x, y and z).
- if u is adjacent to none of $\{s_1, s_2, s_3\}$, then these vertices are still in the reduced graph of the second item above, and, moreover, there is no edge between X_2 and X_3, using similar arguments as previously.

Case 2: c_j is adjacent to X_4 but not to X_2, X_3. In this case, we claim that there exists $j' \neq j$ such that $c_{j'}$ is adjacent to X_1 or X_4 (or both): if this is not the case and the instance is positive, there would be an independent set of size k which intersects $\{c_1, \dots, c_{k-1}\}$ (by considering $\{c_1, \dots, c_{k-1}\} \setminus \{c_j\}$ together with the vertices of the rainbow independent set intersecting X_1 and X_4), which is impossible in an instance of f-Ramsey-extracted Iterative Expansion MIS. However, we may assume that $c_{j'}$ is not adjacent to X_1: since we applied Lemma 33 on the couple $(1,2)$, there exists an edge xy in X_1 and a vertex z in X_2 such that xz is an edge and yz is not, but in this case $\{x, y, z, c_j, c_{j'}\}$ would induce a dart. Hence X_1 must induce an independent set: if it is of size at least k we are done, otherwise we may branch to decide which vertex should be in the solution. Hence, $c_{j'}$ is adjacent to X_4 and not to X_1.

▷ Claim 40. We can FPT-reduce to the case where X_1 induces a clique.

Proof. First observe that the neighborhood in X_1 of any vertex of X_4 must be a clique, for otherwise there would be a dart using c_j and $c_{j'}$. We choose an arbitrary vertex $u \in X_4$. Assume the instance is positive: let $s_1 \in X_1$ and $s_4 \in X_4$ be the elements of a rainbow independent set. Notice that there are three possibilities: either $u = s_4$, or $us_4 \in E(G)$ or $us_4 \notin E(G)$. We perform a branching corresponding to these possibilities: in the first branch u is taken in the solution, and κ decreases. In the second branch we remove $N(u)$ from X_4, and thus the size of a maximum independent set in $G[X_4]$ decreases, and the same holds for κ. In the third branch we remove from X_4 the non-neighbors of u and perform another branching in order to guess whether u is a neighbor of s_1 or not. In the first branch we thus replace X_1 by $X_1 \cap N(u)$ which is a clique, as desired. In the second branch we replace X_1 by $X_1 \setminus N(u)$, but in this case X_1 is now P_3-free: indeed, if a, b, c induces a P_3, then $\{a, b, c, u, c_j\}$ induces a dart. Hence X_1 induces a disjoint union of cliques: if there are at least k of them we are done, and otherwise we branch once again and replace X_1 by a clique. ◁

Let us now summarize the situation: c_j is adjacent to X_1 and X_4 but not to X_2 nor X_3, and X_1 induces a clique. There must be two vertices $c_{j_2}, c_{j_3} \in \{c_1, \dots, c_{k-1}\}$ ($j_2 \neq j_3$) dominating $X_2 \cup X_3$ (otherwise, we would be able to find an independent set of size k intersecting $\{c_1, \dots, c_{k-1}\}$, as previously). If one of them, w.l.o.g. c_{j_2}, is also adjacent to X_1, then we can apply the same argument with c_{j_2} instead of c_j. We are then in Case 1 and we can reduce. So we can assume that c_{j_2} and c_{j_3} are not adjacent to X_1. If both c_{j_2} and c_{j_3} dominate X_2, then we claim that we can branch to decrease the number of edges in the set graph. To do so, we claim that if a vertex x of X_1 is adjacent to a connected component of X_2, it is adjacent to the whole component. Indeed otherwise, there exists an edge yz of X_2 such that xz is an edge and not xy. And then we obtain a dart with universal vertex z using c_{j_1}, y, c_{j_2} and x. So we can branch over the connected components of X_2 to guess in which connected component is selected the vertex of X_2 and then delete the neighbors of this component in X_1. In the resulting graph, the edge set of the set graph decreased.

So from now on we assume that c_{j_2} is adjacent to X_2 and not to X_1, X_3 and c_{j_3} is adjacent to X_3 and not to X_1, X_2 (and we do not assume anything about the adjacencies of c_{j_2} and c_{j_3} to X_4). We say that an edge between X_2 and X_3 is *strong* if it is contained in a triangle using a vertex of X_1. If ab is a strong edge, we claim that $\{a, b\}$ is a module *w.r.t.* X_1: otherwise, let $x \in X_1$ such that abx is a triangle, let $x' \in X_1$ such that ax' is an edge but bx' is not: $\{a, b, x, x', c_{j_2}\}$ induces a dart (remember that xx' is an edge, since X_1 induces a clique). Hence, every connected component C of the subgraph consisting of strong edges is a module *w.r.t.* X_1. In particular, every such connected component C is in the neighborhood of some vertex $x \in X_1$ which is adjacent to c_j (while c_j is not adjacent to any vertex of C). Hence, the (connected) subgraph induced by C must be P_3-free (otherwise, such a P_3 together with x and c_j would induce a dart), it is thus a clique. Both X_2 (resp. X_3) can therefore be partitioned into X_2^1, $\dots$, X_2^p (resp. X_3^1, $\dots$, X_3^p) such that every vertex of X_2^a is connected with strong edges to every vertex of X_3^a, for every $a \in \{1, \dots, p-1\}$, and every vertex of X_2^p (resp. X_3^p) is not incident to any strong edge. We now perform a random reduction : pick at random a non trivial subset A of $[p-1]$, remove from X_2 all vertices from $\cup_{a \in A} X_2^a$ and remove from X_3 all vertices from $\cup_{a \in [p-1] \setminus A} X_3^a$. Assume the instance is positive: let $(s_2, s_3) \in X_2 \times X_3$ be the elements of a rainbow independent set. Since strong edges are edges of G, there exist $a_2, a_3 \in [p]$ such that $(s_2, s_3) \in X_2^{a_2} \times X_3^{a_3}$ with $a_2 \neq a_3$ or $a_2 = a_3 = p$. Hence, the probability that s_2 and s_3 have not been deleted is at least 1/2 (and if the instance is negative, the reduced one is negative as well, since the reduced graph is an induced subgraph of the former). Moreover, in the reduced instance, there is no strong edge between X_2 and X_3 (but, there still might be edges). However, if two vertices x, x' in X_1 have a common neighbor in X_2 (resp. X_3), then they must be twins in X_3 (resp X_2), otherwise we would be able to form a dart together with c_j. This means that all edges between X_1 and X_2 can actually be partitioned into (not necessarily induced) complete bipartite graphs.

We can thus perform a random reduction similar to the previous one, except that in the reduced instance, there will not be any edge between X_1 and X_2. Hence, the parameter κ decreases, which concludes the proof. ◀

From now on, we may assume that the set graph is {claw, paw}-free. In that case, we prove that it has a simple structure: it is either a path, a cycle or the complement of a matching. The first two cases will then be handled in Lemma 42 and the last one in Lemma 34.

▶ **Lemma 41.** *If a connected graph G is {claw, paw}-free, it is either a path, a cycle or the complement of a (not necessarily perfect) matching.*

Proof. By contradiction. Let P be a induced path or cycle of maximal size of G (if a path of the same length exists, we choose the path). First note that if $P \in \{P_2, C_3\}$ then G is a clique and the conclusion holds. So we can assume that P is P_k with $k \geq 3$ or $C_{k'}$ with $k' \geq 4$. We prove that all vertices of $V \setminus P$ are either complete or anti-complete to P. Indeed let x be a vertex adjacent to a vertex of P. If x is adjacent to an internal vertex y of P, then since G is claw-free, it must also be adjacent to a neighbor of y. If it is not complete to P, let y_1, y_2, y_3 be consecutive vertices of P such that x is adjacent to both y_1, y_2 but not y_3. Then x, y_1, y_2, y_3 is a induced paw since $P \neq C_3$. If x is not adjacent to the interior of P, we can either increase the length of the path or create a cycle with an additional vertex, a contradiction. Since G is connected, all the vertices that are not adjacent to P have to be adjacent to a vertex dominating P, which creates a paw. So such vertices do not exist.

Now remark that if the set of dominating vertices is not empty, then P is either a P_3 or a C_4 since otherwise there is a paw. Also note that every vertex of $V \setminus P$ has co-degree at most one in $V \setminus P$ since otherwise there is a paw or a claw using a vertex of P. So $V \setminus P$ induces the complement of a (not necessarily perfect) matching, and is complete with either P_3 or C_4, which are themselves the complement of a matching. ◀

Because of the previous result and Lemma 34, the only remaining case is where the set graph is the complement of a (non necessarily perfect) matching.

▶ **Lemma 42** (Particular set graph: complement of matching). *If the set graph is the complement of a (not necessarily perfect) matching, then we can output $O(h(k)|V(G)|^2)$ instances in which the set graph is bipartite, for some computable function h.*

Proof. Assume w.l.o.g. that X_1 is a vertex of the set graph of degree at least two in $\{c_1, \ldots, c_{k-1}\}$. Such a vertex exists since the bipartite graph between the sets X_1 and $\{c_1, \ldots, c_{k-1}\}$ is connected. Let c_j and $c_{j'}$ be two neighbors of X_1. We apply Lemma 33 to the set of couples $\{(i,j) : i < j\}$, and now claim that the instances we obtain are FPT (without applying any new FPT-reduction). First observe that for every $r \neq 1$ such that X_r is adjacent to X_1, X_1 is adjacent to at least one of $\{c_j, c_{j'}\}$. Indeed, otherwise, pick an edge xy in X_1 together with a vertex z in X_r such that xz is an edge and yz is not (this configuration is possible since all couples $(1, r)$ are clean for all $r \neq 1$): $\{x, y, z, c_j, c_{j'}\}$ induces a dart. Hence, $\{c_j, c_{j'}\}$ is adjacent to all the vertices of the set graph but at most one (since X_1 has at most one non-neighbor in the set graph).

Assume w.l.o.g. that X_k is the unique (if it exists) non-neighbor of X_1. We now try all possible choices for $s_1 \in X_1$ and $s_k \in X_k$ (if X_k does not exist, we just try all choices for s_1), and for every $\ell \neq 1, k$, we remove from X_ℓ all neighbors of s_1 and s_k. If some of these sets become empty, then we can answer "No". Otherwise, observe that every remaining vertex is in the neighborhood of either c_j or $c_{j'}$ and in the non-neighborhood of s_1, and s_1 is a neighbor of c_j and $c_{j'}$. Hence, the neighborhood of c_j (resp. $c_{j'}$) in the remaining vertices must be P_3-free. Thus, all remaining vertices can be partitioned into two disjoint union of cliques $A_1, \ldots, A_p$ and $B_1, \ldots, B_q$. If $p \geq k-2$ or $q \geq k-2$, then we are done ($p \geq k-1$ or $q \geq k-1$ if X_k does not exist). Otherwise, we branch in order to guess which of these cliques contain a solution of a rainbow independent set. The remaining cliques now play the role of the sets X_ℓ, and we thus end up with $O(4^k n^2)$ instances of f-Ramsey-extracted Iterative Expansion MIS whose set graph is a bipartite graph, as desired. ◀

Observe that all FPT branchings of our algorithm either end on the case where the set graph is a path or a cycle, in which case we conclude by Lemma 34, or where the set graph is the complement of a matching, in which case we have $O(h(k)n^2)$ instances where the set graph is bipartite. But in the latter case, since every new branching only removes edges of the set graph, it must remain bipartite, which ensures that the branching of Lemma 42 will be performed only once, except if the set graph is both the complement of a matching and bipartite, but in this case it must be of size at most 4, and we conclude by an exhaustive guess instead of applying the above lemma. ◀

5.2 The cricket

▶ **Theorem 43.** *There is a randomized FPT algorithm for* MIS *in cricket-free graphs parameterized by the size of the solution.*

Proof. Let us first prove that we can assume some additional information on the graph G.

▶ **Lemma 44.** *If G has a dominating set of constant size, then we can solve* MIS *in FPT time.*

Proof. Let $D \subseteq V(G)$ be such a dominating set (it can be found in $O(|E(G)| \cdot |V(G)|^{|D|})$ time). Let us partition the vertices of $V(G) \setminus D$ into at most $2^{|D|}$ sets $S_1, \ldots, S_p$, depending on their exact neighborhood in D. Since G is cricket-free, $G[S_i]$ is O_4-free, where O_4 is the graph with four vertices and one edge. We branch in order to guess whether the solution has an intersection with S_i of size 0, 1 or at least 2, for every $i \in [p]$. Observe that there are at most $3^{2^{|D|}}$ such choices. Consider one of these choices. We delete sets with intersection of size 0, and guess one vertex in each set with intersection 1 and decrease k by one. For every set S_i with intersection at least 2, we guess two vertices and remove their neighborhood. Since $G[S_i]$ is O_4-free graphs, the remaining vertices must form an independent set. If one of them is of size at least k we are done. Otherwise, the remaining vertices are of size at most $k \cdot 2^{|D|}$, and we can conclude by brute-force. ◀

▶ **Lemma 45.** *We can reduce to the case where G is $K_{2,3}$-free.*

Proof. We prove it by induction on k. Let C be a $K_{2,3}$. If $|N(C)|$ is bounded by a function of k, we branch to guess which vertex in $N[C]$ has to be selected in the solution and we decrease the invariant (indeed, for any set of vertices Q, any maximal independent set must intersect either Q or $N(Q)$). So from now on, we assume that $N[C]$ is arbitrarily large. Let us denote by X the independent set of size 2 in C, and by Y the independent set of size 3. We first consider the set Z of vertices z only adjacent to X but not to Y or to Y but not to X. We claim that there are at most $5k$ such vertices: for every $x \in C$, we may assume $|N(x) \cap Z| \geq k$, for otherwise either $N(x) \cap Z$ induces an independent set (in which case we are done), or it contains an edge, in which case this edge together with x and two neighbors of x in C induce a cricket.

Let A be the set of vertices adjacent to both sides of C. We claim that every $a \in A$ is adjacent to at least two vertices of Y. Indeed, otherwise $\{a\}$ union Y union a vertex of X adjacent to a induces a cricket. In particular, every vertex adjacent to C is adjacent to an edge and a non-edge of C. Let $a \in A$. All but at most one neighbor of a is adjacent to C. Indeed, otherwise a has two neighbors u, v non-adjacent to C. If uv is an edge, u, v, a and a non-edge of $N(a) \cap C$ induce a cricket. If uv is a non-edge, then a, u, v and an edge of $N(a) \cap C$ induce a cricket.

Let $U := C \cup A$ and $W := V \setminus (C \cup A)$. Every vertex of U is adjacent to at most $5k + 1$ vertices of W, namely vertices of Z plus at most one vertex. So U is one-sided almost disconnected to W. If W is of size bounded by a function of k, notice that A is dominated by two vertices of Y, in which case we conclude by Lemma 44. So by induction and by Lemma 18, the problem can be decided in FPT time. ◀

▶ **Lemma 46.** *We can reduce to the case where G is $K_{1,5}$-free.*

Proof. By Lemma 45, we may assume that G is $K_{2,3}$-free. Let C be an induced $K_{1,5}$ and let us denote by v the center of the star and $v_1, \dots, v_5$ its leaves.

First note that there are at most k vertices A adjacent to v but not adjacent to $\{v_1, \dots, v_5\}$. Indeed, otherwise A would contain an edge, and this edge, v and v_1, v_2 would induce a cricket. If a vertex x is adjacent to v and at least one v_i, then at most one v_j is not in $N(x)$. Indeed otherwise x, v, v_i and two non-neighbors of x in $\{v_1, \dots, v_5\}$ would form a cricket. Note moreover that no vertex can be adjacent to three leaves of the star but not to v since otherwise there would be a $K_{2,3}$ in G, a contradiction with Lemma 45. So $V \setminus C$ can be partitioned into the following sets: $X_i, Y_i, Y_{i,j}, X, Y$ where $1 \leq i, j \leq 5$. The set X_i denotes the set of vertices whose neighborhood in C is v union all leaves but v_i. The set Y_i (resp. $Y_{i,j}$ is the set of vertices whose neighborhood in C is v_i (resp. $\{v_i, v_j\}$). And X (resp. Y) is the set of vertices complete (resp. anti-complete to) C.

The set $X \cup X_i$ is anti-complete to the sets $Y_{\ell,j}$ for all i, j, Y and Y_j for $j \neq i$. Indeed otherwise we create a triangle using this edge plus a common neighbor on C and complete the cricket with two neighbors of the vertex of $X \cup X_i$ that are not neighbors of the vertex of Y_j (resp. $Y_{\ell,j}$). Moreover, every vertex $x \in X \cup X_i$ has at most one neighbor in $Y_i \cup Y$. Indeed, if there is a non-edge in $N(x) \cap (Y_i \cup Y)$, we complete the cricket with the edge v, v_j with $j \neq i$ adjacent to x. If there is an edge in $N(x) \cap (Y_i \cup Y)$, we complete the cricket with the non-edge v_j, v'_j with $j, j' \neq i$ adjacent to x.

So every vertex of $U = C \cup X \cup_{i \leq 5} X_i$ has degree at most $k + 1$ in $W = \bigcup_{i \leq 5} Y_i \bigcup_{i,j \leq 5} Y_{i,j} \cup Y \cup A$. Note that U is dominated by v and thus, if $|W|$ is bounded by a function of k, we can conclude by branching on W and applying Lemma 44. Otherwise, by induction and by Lemma 18, the problem can be decided in FPT time. ◀

In order to prove that MIS is (randomized) FPT in cricket-free graphs, we first apply Lemmas 45 and 46 above. These lemmas ensure that in order to prove our result, it is sufficient to give a (randomized) FPT algorithm for MIS in $\{cricket, K_{2,3}, K_{1,5}\}$-free graphs. To do so, we now prove that f-Ramsey-extracted Iterative Expansion MIS is (randomized) FPT in $\{cricket, K_{2,3}, K_{1,5}\}$-free graphs, where $f(x) = 2$ for all $x \in \mathbb{N}$. That is, $V(G)$ is partitioned into $X_1 \cup \dots \cup X_k \cup C_1 \cup \dots C_{k-1}$, and, for every $j \in [k-1]$, we have $C_j = \{c_j^1, c_j^2\}$.

In the remainder of the proof, R denotes the set graph.

▶ **Lemma 47.** *Let X_iX_j be an edge of the set graph. If all the vertices of X_i have degree at most k in X_j, we can reduce.*

Proof. Assume that all the vertices of X_i have degree at most k on X_j. We now use a random reduction. Namely, we delete every vertex of X_j with probability $1/2$. We then delete all vertices x of X_i having a neighbor in X_j. After this transformation, there is no edge anymore between X_i and X_j and then the invariant has decreased. Moreover, if the instance is positive, then the vertex of X_j in the rainbow solution is still in X_j with probability $1/2$ and the one of X_i is still in X_i with probability at least $1/2^k$, which completes the proof. ◀

We consider two types of sets X_i depending of the number of elements of $\{c_1^1, \dots, c_{k-1}^1\}$ they see. A vertex X_i of the set graph is *type 1* if it sees only one element of this set, otherwise it is *type 2*.

▶ **Lemma 48.** *In a positive instance, every type 2 vertex of the set graph has degree at most* 6*.*

Proof. Assume that X_i is of type 2, and let c^1_j, $c^1_{j'}$ be adjacent to X_i. Since G is $K_{1,5}$-free by Lemma 46, the vertex c^1_j (resp. $c^1_{j'}$) is adjacent to at most three X_r's distinct from X_i (since, if the instance is positive, five elements of a rainbow independent set would induce a $K_{1,5}$). Hence, if X_i has degree at least 7 in the set graph, it must be adjacent to some X_r which is not itself adjacent to c^1_j nor $c^1_{j'}$. If there is a vertex x in X_i of degree at least k in X_r, then either these vertices are independent, in which case we are done, or there is an edge ab. But in that case $\{a, b, x, c^1_j, c^1_{j'}\}$ induces a cricket. Hence the degree of every vertex of X_i in X_r is at most k, in which case we can reduce by Lemma 47. ◀

▶ **Lemma 49.** *We can reduce so that the set graph does not contain any triangle of type* 1 *vertices.*

Proof. W.l.o.g., assume X_1, X_2, X_3 is a triangle, each vertex respectively dominated by C_1, C_2, C_3. Note that if C_i and C_j with $i \neq j \in \{1, 2, 3\}$ are not distinct then we can replace c^1_i by an independent set of size two in $X_i \cup X_j$ and obtain an independent set of size k intersecting $C_1 \cup \cdots \cup C_{k-1}$, which is impossible in an instance of f-Ramsey-extracted Iterative Expansion MIS. So from now on, we will assume that C_1, C_2, C_3 are pairwise distinct.

Now, observe that every vertex x of X_1 has a neighborhood in $X_2 \cup X_3$ which is a clique, otherwise there is a cricket. By symmetry the same holds for X_2 and X_3. So the tripartite graph on X_1, X_2, X_3 is a disjoint union of complete tripartite graphs, and we can conclude with a random reduction. Namely, for each component T of the complete tripartite graph between X_1, X_2, X_3, we choose to keep $T \cap X_1$ with probability 1/3, or to keep $T \cap X_2$ with probability 1/3 or to keep $T \cap X_3$ with probability 1/3. With probability at least 1/27, the vertices of the rainbow independent set in X_1, X_2, X_3 are still in the resulting graph. Moreover, after this operation, X_1, X_2, X_3 is an independent set of the set graph, so the number of edges in the set graph decreased. ◀

▶ **Lemma 50.** *The set graph R is claw-free.*

Proof. Assume that a vertex X_i of R has an independent set of size 3 in its neighborhood, namely X_1, X_2, X_3. By Lemma 47, there exists a vertex x of X_i with at least k neighbors in X_1. So there is an edge in its neighborhood in X_1. By completing this set with a neighbor of X_i in X_2 and X_3, we obtain a cricket. ◀

▶ **Lemma 51.** *The maximum degree in the set graph R is* 14*.*

Proof. Assume that some X_i has degree at least 15. By Lemma 48, it must be of type 1. By Lemma 49, the neighborhood of X_i cannot contain two adjacent type 1 vertices. And it cannot contain three non adjacent vertices by Lemma 50. So it has at most 2 type-1 neighbors. Now if X_i has 13 neighbors $\mathcal{X}$ of type 2, since each of them have maximum degree at most 6 by Lemma 49 (including type 1 neighbors), the subgraph induced by $\mathcal{X}$ contains an independent set of size 3, a contradiction with Lemma 50. ◀

▶ **Lemma 52.** *We can reduce so that the set graph R is bull-free.*

Proof. Let X_1 be the chin of the bull, and X_2, X_3 be the non horns and X_4, X_5 be the horns adjacent to respectively X_2 and X_3. Using an FPT branching, we show that we may assume that for every vertex of x of X_2 we have that $N(x) \cap X_4$ contains an edge. Indeed, for a given rainbow independent set, there are two cases: either the vertex of X_2 has degree at most k in X_4, or it has degree at least $k + 1$ in X_4. For the first branch, we remove from X_2 all vertices with degree greater than k, and, by Lemma 47, we can reduce. We thus end up with the second branch, where every vertex of X_2 as degree at least $k + 1$ in X_4. Since these neighbors are not an independent set (otherwise we are done), they must induce an edge. Similarly, we may assume that for every vertex y in X_3, we have that $N(y) \cap X_5$ contains an edge.

First note that, for every edge xy between X_2, X_3 $N(x) \cap X_1 = N(y) \cap X_1$. Indeed otherwise, we can assume by symmetry that there exists $z \in X_1$ adjacent to x but not to y. Then, using an edge in the neighborhood of x in the horn, we make a cricket.

Using the same idea as previously, we may also assume that every vertex of X_2 has degree at least k in X_3.

We distinguish two cases, which correspond to two branches. First assume that the bipartite graph B between X_2 and X_3 (*i.e.* not taking into account the edges induced by X_2 nor X_3) is connected. Then all the vertices of X_2, X_3 have exactly the same neighborhood in X_1. Since we are looking for an independent set containing a vertex of X_1, X_2 and X_3, all the neighbors of vertices of X_2 in X_1 can be deleted. After this modification, there is no edge anymore between X_1 and X_2 nor between X_1 and X_3.

So we may assume that the bipartite graph B between X_2 and X_3 is not connected. If the solution does not select a vertex in X_2 and in X_3 in the same component of B, then we can conclude using a random reduction: for each connected component T of the bipartite graph, we keep $T \cap X_2$ with probability 1/2 or we keep $T \cap X_3$ with probability 1/2. The vertices of the rainbow independent set in X_2 and X_3 are still in the graph with probability 1/4. After this branching, the number of edges in the set graph decreases.

So we may finally assume that the solution selects a vertex in X_2 and X_3 in the same connected component of B. Remark that, for every edge $x, x' \in X_2$ where x, x' lie in distinct connected components of B, then x, x' have the same neighborhood in X_4. Indeed, otherwise we can assume w.l.o.g. that there is a vertex $y \in X_4$ adjacent to x but not to x'. Then x, x', y plus an edge of $N(x) \cap X_3$ induces a cricket.

Let us denote by H the subgraph of $G[X_2]$ where xy is an edge if xy is an edge of G and x, y are not in the same component of B. We now run the following algorithm: we start with $S = \emptyset$ and $T = \emptyset$ and $W = \emptyset$. As long as there remains a component B' of B and C' of H such that $B' \notin S$ and $C' \notin T$ and there exists $x \in B' \cap C'$, we add B in S and C in T and $x \in W$. We repeat this operation as long as we can. We claim that W is an independent set of G. Indeed assume by contradiction that xx' is an edge and that x' is added in W after x. The vertex x' is not in the connected component of x in B by definition of S; and it is not in the connected component of x in H by definition of T. Since every edge of $G[X_2]$ that is not in H is in $X_2 \cap B'$ for some component B' of B, we have a contradiction. So finally all the vertices of X_2 are in the connected component of S in B or in the connected component T in H. We branch over all the possible choices in order to guess in which connected component of S and T the vertex of X_2 in a rainbow solution lies (in each branch, we replace X_2 by the corresponding connected component). Clearly there at most $2k$ choices. In the resulting branchings for S, all the vertices of X_2 now have the same neighborhood in X_1 and then we can reduce the number of edges in the set graph. And in the resulting branchings for T, all the vertices of X_2 now have the same neighborhood in X_4 and then we can reduce the number of edges in the set graph, which completes the proof. ◀

▶ **Lemma 53.** *We can solve the reduced instance in polynomial time.*

Proof. The idea is to prove that the pathwidth of the set graph is bounded by some constant, and then apply a dynamic programming similar to the ones of Lemma 34 where the set graph is a path.

Consider a longest induced path P in the set graph R. Since the maximum degree of the set graph is bounded by Lemma 51 and that R is connected (and since we can directly conclude if R is small enough), we can assume that P has length at least 14.

Note that P dominates R. Indeed let X be a vertex of R not adjacent to P. Since R is connected, we can assume that R is at distance two from P and let Y be a neighbor of X adjacent to P. If Y is only adjacent to an endpoint of P, P is not maximal. If it is adjacent to both endpoints but not the internal vertices, there is claw, a contradiction with Lemma 50. If it is adjacent to an internal vertex (but not its neighbors), there is a claw, a contradiction with Lemma 50. If it is adjacent to some (but not all) vertices of P, then there is a bull, a contradiction with Lemma 52. So P dominates R.

Let Y be a vertex in the neighborhood of P. We claim that either Y sees an endpoint of P or that its neighborhood in P consists of at most 4 consecutive vertices. Indeed, assume that Y is not connected to an endpoint of P and let X_i be its rightmost neighbor on the path P. If Y is not adjacent to X_{i-1}, the vertex before X_i in P, there is a claw, a contradiction with Lemma 50. If it is adjacent to X_{i-1} but not

X_{i-2}, there is a bull, a contradiction with Lemma 52. So Y has to be adjacent to all of X_i, X_{i-1}, X_{i-2}. But then if it is adjacent a vertex X_j with $j \leq i-4$, then there is a claw, a contradiction with Lemma 50.

It implies that the set graph R has pathwidth at most 85. Indeed, let us consider a path $p_1, \dots, p_\ell$ of length $\ell := |P|$. Every vertex of R adjacent to an endpoint of P is added in all the bags. For any other vertex $X_i \notin P$ whose first neighbor on P is p_j, we add X_i in the bags of $p_j, \dots, p_{j+3}$. We finally add the vertex X_i of P in the p_i an p_{i-1}. All bags contain at most $6 \cdot 14 + 2 = 86$ sets X_i by Lemma 51. So the pathwidth of the set graph is at most 85. Note moreover that this decomposition can be found in polynomial time (we start with a maximal path by inclusion and either we improve it or we find the decomposition).

So we can now find an independent set in polynomial time using dynamic programming. We order the sets X_i in such a way that if $X_j \succ X_i$ the vertex X_j first appear in a bag later than X_i. We now claim that choosing a vertex in X_{i+1} knowing $X_1, \dots, X_i$ is equivalent to choosing a vertex X_{i+1} only knowing $(\cup_{j \leq i} X_i) \cap \mathcal{B}$ where $\mathcal{B}$ is the first bag of the path decomposition containing X_{i+1}. Indeed, by definition of path decomposition, if X_j with $j \leq i$ is not in $\mathcal{B}$, then there is no edge in the set graph between $X_i X_j$ (since X_j only appears in a subpath of P and has already disappeared). So in order to decide which further vertices can be selected, we only need to keep track of the vertices selected in the current bag $\mathcal{B}$ of the path decomposition. Since the size of the bags is bounded, we obtain a polynomial time dynamic programming algorithm to decide the problem in that case. ◀

That finishes the proof. ◀

References

1 Vladimir E. Alekseev. The effect of local constraints on the complexity of determination of the graph independence number. *Combinatorial-Algebraic Methods in Applied Mathematics*, pages 3–13, 1982. in Russian.

2 Vladimir E. Alekseev. Polynomial algorithm for finding the largest independent sets in graphs without forks. *Discrete Applied Mathematics*, 135(1-3):3–16, 2004. URL: `https://doi.org/10.1016/S0166-218X(02)00290-1`, `doi:10.1016/S0166-218X(02)00290-1`.

3 Gábor Bacsó, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Zsolt Tuza, and Erik Jan van Leeuwen. Subexponential-time algorithms for maximum independent set in P_t-free and broom-free graphs. *Algorithmica*, 81(2):421–438, 2019. URL: `https://doi.org/10.1007/s00453-018-0479-5`, `doi:10.1007/s00453-018-0479-5`.

4 Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Stéphan Thomassé, and Rémi Watrigant. Parameterized complexity of independent set in H-free graphs. In *13th International Symposium on Parameterized and Exact Computation, IPEC 2018, August 20-24, 2018, Helsinki, Finland*, pages 17:1–17:13, 2018. URL: `http://dx.doi.org/10.4230/LIPIcs.IPEC.2018.17`, `doi:10.4230/LIPIcs.IPEC.2018.17`.

5 Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Stéphan Thomassé, and Rémi Watrigant. Parameterized complexity of independent set in H-free graphs. *CoRR*, abs/1810.04620, 2018.

6 Andreas Brandstädt and Raffaele Mosca. Maximum weight independent set for ℓclaw-free graphs in polynomial time. *Discrete Applied Mathematics*, 237:57–64, 2018. URL: `https://doi.org/10.1016/j.dam.2017.11.029`, `doi:10.1016/j.dam.2017.11.029`.

7 Andreas Brandstädt and Raffaele Mosca. Maximum weight independent sets for (P_7, triangle)-free graphs in polynomial time. *Discrete Applied Mathematics*, 236:57–65, 2018. URL: `https://doi.org/10.1016/j.dam.2017.10.003`, `doi:10.1016/j.dam.2017.10.003`.

8 Andreas Brandstädt and Raffaele Mosca. Maximum weight independent sets for ($S_{1,2,4}$, triangle)-free graphs in polynomial time. *CoRR*, abs/1806.09472, 2018. URL: `http://arxiv.org/abs/1806.09472`, `arXiv:1806.09472`.

9 Jianer Chen, Yang Liu, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Iterative expansion and color coding: An improved algorithm for 3D-matching. *ACM Trans. Algorithms*, 8(1):6:1–6:22, 2012.

10 Derek G. Corneil, Yehoshua Perl, and Lorna K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14(4):926–934, 1985.

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

12 Konrad Dabrowski. *Structural Solutions to Maximum Independent Set and Related Problems*. PhD thesis, University of Warwick, 2012.

13 Konrad Dabrowski, Vadim V. Lozin, Haiko Müller, and Dieter Rautenbach. Parameterized complexity of the weighted independent set problem beyond graphs of bounded clique number. *J. Discrete Algorithms*, 14:207–213, 2012.

14 Konrad K. Dabrowski, Vadim V. Lozin, Dominique de Werra, and Victor Zamaraev. Combinatorics and algorithms for augmenting graphs. *Graphs and Combinatorics*, 32(4):1339–1352, 2016. URL: `https://doi.org/10.1007/s00373-015-1660-0`, `doi:10.1007/s00373-015-1660-0`.

15 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

16 Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

17 Henri Perret du Cray and Ignasi Sau. Improved FPT algorithms for weighted independent set in bull-free graphs. *Discrete Mathematics*, 341(2):451–462, 2018.

18 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

19 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *J. ACM*, 64(3):17:1–17:32, 2017. URL: `https://doi.org/10.1145/3051095`, `doi:10.1145/3051095`.

20 Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. URL: `https://doi.org/10.1007/BF02579273`, `doi:10.1007/BF02579273`.

21 Andrzej Grzesik, Tereza Klimosova, Marcin Pilipczuk, and Michal Pilipczuk. Polynomial-time algorithm for maximum weight independent set on P_6-free graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1257–1271, 2019. URL: `https://doi.org/10.1137/1.9781611975482.77`, `doi:10.1137/1.9781611975482.77`.

22 Ararat Harutyunyan, Michael Lampis, Vadim V. Lozin, and Jérôme Monnot. Maximum independent sets in subcubic graphs: New results. *CoRR*, abs/1810.10940, 2018. URL: `http://arxiv.org/abs/1810.10940`, `arXiv:1810.10940`.

23 Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 627–636, 1996. URL: `https://doi.org/10.1109/SFCS.1996.548522`, `doi:10.1109/SFCS.1996.548522`.

24 Tamás Kovári, Vera Sós, and Pál Turán. On a problem of K. Zarankiewicz. In *Colloquium Mathematicum*, volume 1, pages 50–57, 1954.

25 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: `http://eatcs.org/beatcs/index.php/beatcs/article/view/92`.

26 Daniel Lokshtanov, Marcin Pilipczuk, and Erik Jan van Leeuwen. Independence and efficient domination on P_6-free graphs. *ACM Trans. Algorithms*, 14(1):3:1–3:30, 2018. URL: `https://doi.org/10.1145/3147214`, `doi:10.1145/3147214`.

27 Daniel Lokshtanov, Martin Vatshelle, and Yngve Villanger. Independent set in P_5-free graphs in polynomial time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 570–581, 2014.

28 Vadim V. Lozin. From matchings to independent sets. *Discrete Applied Mathematics*, 231:4–14, 2017. URL: `https://doi.org/10.1016/j.dam.2016.04.012`, `doi:10.1016/j.dam.2016.04.012`.

29 Vadim V. Lozin and Martin Milanič. Maximum independent sets in graphs of low degree. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 874–880, 2007. URL: `http://dl.acm.org/citation.cfm?id=1283383.1283477`.

30 Vadim V. Lozin and Martin Milanič. A polynomial algorithm to find an independent set of maximum weight in a fork-free graph. *J. Discrete Algorithms*, 6(4):595–604, 2008. URL: https://doi.org/10.1016/j.jda.2008.04.001, doi:10.1016/j.jda.2008.04.001.

31 Vadim V. Lozin, Jérôme Monnot, and Bernard Ries. On the maximum independent set problem in subclasses of subcubic graphs. *J. Discrete Algorithms*, 31:104–112, 2015. URL: https://doi.org/10.1016/j.jda.2014.08.005, doi:10.1016/j.jda.2014.08.005.

32 Frédéric Maffray and Lucas Pastor. Maximum weight stable set in (P_7, bull)-free graphs and ($S_{1,2,3}$, bull)-free graphs. *Discrete Mathematics*, 341(5):1449–1458, 2018. URL: https://doi.org/10.1016/j.disc.2017.10.004, doi:10.1016/j.disc.2017.10.004.

33 Dmitriy S. Malyshev. Classes of subcubic planar graphs for which the independent set problem is polynomially solvable. *Journal of Applied and Industrial Mathematics*, 7(4):537, 2013.

34 Dmitriy S. Malyshev and Dmitrii V. Sirotkin. Polynomial-time solvability of the independent set problem in a certain class of subcubic planar graphs. *Journal of Applied and Industrial Mathematics*, 11(3):400–414, 2017.

35 George J. Minty. On maximal independent sets of vertices in claw-free graphs. *J. Comb. Theory, Ser. B*, 28(3):284–304, 1980. URL: https://doi.org/10.1016/0095-8956(80)90074-X, doi:10.1016/0095-8956(80)90074-X.

36 Najiba Sbihi. Algorithme de recherche d'un stable de cardinalite maximum dans un graphe sans etoile. *Discrete Mathematics*, 29(1):53–76, 1980. URL: https://doi.org/10.1016/0012-365X(90)90287-R, doi:10.1016/0012-365X(90)90287-R.

37 Stéphan Thomassé, Nicolas Trotignon, and Kristina Vuskovic. A polynomial turing-kernel for weighted independent set in bull-free graphs. *Algorithmica*, 77(3):619–641, 2017.

38 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.