Parameterized Complexity Dichotomy for $(r,\ell)$-Vertex Deletion

Julien Baste$^1$, Luerbio Faria$^2$, Sulamita Klein$^3$, and Ignasi Sau$^1$

$^1$ AlGCo project team, CNRS, LIRMM, Montpellier, France. 
julien.baste@lirmm.fr, ignasi.sau@lirmm.fr

$^2$ FFP, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil. 
luerbio@cos.ufrj.br

$^3$ Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil. 
sula@cos.ufrj.br

Abstract. For two integers $r,\ell \geq 0$, a graph $G = (V, E)$ is an $(r,\ell)$-graph if $V$ can be partitioned into $r$ independent sets and $\ell$ cliques. In the parameterized $(r,\ell)$-Vertex Deletion problem, given a graph $G$ and an integer $k$, one has to decide whether at most $k$ vertices can be removed from $G$ to obtain an $(r,\ell)$-graph. This problem is NP-hard if $r+\ell \geq 1$ and encompasses several relevant problems such as Vertex Cover and Odd Cycle Transversal. The parameterized complexity of $(r,\ell)$-Vertex Deletion was known for all values of $(r,\ell)$ except for $(2,1)$, $(1,2)$, and $(2,2)$. We prove that each of these three cases is FPT and, furthermore, solvable in single-exponential time, which is asymptotically optimal in terms of $k$. We consider as well the version of $(r,\ell)$-Vertex Deletion where the set of vertices to be removed has to induce an independent set, and provide also a parameterized complexity dichotomy for this problem.

Keywords: graph modification problem; parameterized complexity; iterative compression; FPT-algorithm; single-exponential algorithm.

1 Introduction

Motivation. Let $r,\ell \geq 0$ be two fixed integers. A graph $G = (V, E)$ is an $(r,\ell)$-graph if $V$ can be partitioned into $r$ independent sets and $\ell$ cliques. In the parameterized $(r,\ell)$-Vertex Deletion problem, we are given a graph $G$ and an integer $k$, and the task is to decide whether at most $k$ vertices can be removed from $G$ so that the resulting graph is an $(r,\ell)$-graph. The optimization version of this problem is known to be NP-hard for $r+\ell \geq 1$ by a classical result of Lewis and Yannakakis [14]. The $(r,\ell)$-Vertex Deletion problem has a big expressive power as it captures several relevant problems for particular cases of the pair $(r,\ell)$. Indeed, for instance, the case $(1,0)$ corresponds to Vertex Cover, the case $(2,0)$ to Odd Cycle Transversal, the case $(1,1)$ to Split Vertex Deletion, and the case $(3,0)$ to whether at most $k$ vertices can be removed so that the resulting graph is 3-colorable.

In this article we are interested in the parameterized complexity of $(r,\ell)$-Vertex Deletion; see [5,7,18] for introductory textbooks to the field. We just recall that a problem defined on an $n$-vertex graph is fixed-parameter tractable
(FPT for short) with respect to a parameter $k$ if it can be solved in \textsf{FPT-time}, i.e., in time $f(k) \cdot n^{O(1)}$. An \textsf{FPT}-algorithm that runs in time $2^{O(k)} \cdot n^{O(1)}$ is called \textit{single-exponential}. For the case of \textsc{Vertex Cover} (VC for short), a simple branching algorithm yields an \textsf{FPT}-algorithm in time $f(k) \cdot n^{O(1)}$. The current fastest algorithm \cite{3} runs in time $2^{tk} \cdot n^{O(1)}$. For the case of \textsc{Odd Cycle Transversal} (OCT for short), the problem was not known to be \textsf{FPT} until Reed \textit{et al.} \cite{19} introduced the celebrated technique of \textit{iterative compression} and solved OCT in time $3^{k} \cdot n^{O(1)}$. The current fastest algorithm \cite{15} uses linear programming and runs in time $2^{31k} \cdot n^{O(1)}$. The \textsc{Split Vertex Deletion} problem can be easily seen to be solvable in single-exponential time since split graphs can be characterized by a finite set of forbidden induced subgraphs \cite{2, 8}. The current fastest algorithm \cite{9} runs in time $2^{k} \cdot n^{O(1)}$ and also uses iterative compression, improving the previously fastest algorithm \cite{15} that uses linear programming and runs in time $2^{31k} \cdot n^{O(1)}$. (See also \cite{13} for parameterized algorithms for \((r, \ell)\)-\textsc{Vertex Deletion} on perfect graphs.)

Note that solving \((r, \ell)\)-\textsc{Vertex Deletion} on a graph $G$ is equivalent to solving \((\ell, r)\)-\textsc{Vertex Deletion} on the complement of $G$. This observation implies that the case $(0, 2)$ can also be solved in time $2^{31k} \cdot n^{O(1)}$. Note also that if $\max\{r, \ell\} \geq 3$, then \((r, \ell)\)-\textsc{Vertex Deletion} is para-\textsf{NP}-complete, hence unlikely to be \textsf{FPT}, as for $k = 0$ the problem corresponds to the recognition of $(r, \ell)$-graphs, which is \textsf{NP}-complete if and only if $\max\{r, \ell\} \geq 3$ \cite{14}.

Therefore, concerning the parameterized complexity of the $(r, \ell)$-\textsc{Vertex Deletion} problem on general graphs, from the above discussion it follows that the only open cases are $(2, 1)$, $(1, 2)$, and $(2, 2)$. Note also that all the cases that are known to be \textsf{FPT} can be solved in single-exponential time.

\textbf{Our results.} In this article we prove that each of the above three open cases is \textsf{FPT} and can also be solved in single-exponential time, thus completely settling the parameterized complexity of $(r, \ell)$-\textsc{Vertex Deletion}. That is, excluding the trivial case where $r + \ell = 0$, we obtain the following dichotomy: the problem is \textsf{FPT} and solvable in single-exponential time if $\max\{r, \ell\} \leq 2$, and para-\textsf{NP}-complete otherwise. As discussed later, a single-exponential running time is asymptotically best possible in terms of $k$ unless the Exponential Time Hypothesis (ETH) fails. A summary of the parameterized complexity of $(r, \ell)$-\textsc{Vertex Deletion} is shown in Table \ref{table:1}, where for each value of $(r, \ell)$, the name of the problem (if any), the function $f(k)$, and the appropriate references are given. We denote by $\overline{\textsc{VC}}$ and $\overline{\textsc{OCT}}$ the complementary problems of VC and OCT, respectively. The results of this article correspond to the gray boxes, ‘p-\textsf{NP}-c’ stands for ‘para-\textsf{NP}-complete’, and ‘\textsf{P}’ means that the corresponding problem is polynomial-time solvable.

We also consider the version of $(r, \ell)$-\textsc{Vertex Deletion} where the set $S$ of at most $k$ vertices to be removed has to further satisfy that $G[S]$ is an independent set. We call this problem \textsc{Independent} $(r, \ell)$-\textsc{Vertex Deletion}. Note that, in contrast to $(r, \ell)$-\textsc{Vertex Deletion}, the cases $(r, \ell)$ and $(\ell, r)$ may not be symmetric anymore. This problem has received few attention in the literature and, excluding the most simple cases, to the best of our knowledge only the case
Parameterized Complexity Dichotomy for \((r, \ell)-\text{Vertex Deletion}\)

\[(2, 0)\] has been studied by Marx et al. \cite{17}, who proved it to be FPT. Note that it also holds that the problem is para-NP-complete if \(\max\{r, \ell\} \geq 3\).

We manage to provide a complete characterization of the parameterized complexity of \textsc{Independent} \((r, \ell)-\text{Vertex Deletion}\). The complexity landscape turns out to be richer than the one for \((r, \ell)-\text{Vertex Deletion}\), and one should rather speak about a trichotomy: the problem is polynomial-time solvable if \(r \leq 1\) and \(\ell \leq 2\), NP-hard and FPT if \(r = 2\) and \(\ell \leq 2\), and para-NP-complete otherwise. A summary of the complexity of \textsc{Independent} \((r, \ell)-\text{Vertex Deletion}\) is shown in Table \(\text{2}\) where our results correspond to the gray boxes. We would like to note that some of the polynomial cases, such as the case \((1, 0)\), are not difficult to prove and may be already known, although we are not aware of it.

**Our techniques.** As most of the previous work mentioned before, our algorithms for \((r, \ell)-\text{Vertex Deletion}\) (Section \(3\)) are based on iterative compression, and are not complicated. (For completeness, we include in Appendix \(A\) some well-known properties of iterative compression.) As a crucial ingredient in our algorithms, we prove (Lemma \(1\) in Section \(2\)) that an \(n\)-vertex \((r, \ell)\)-graph has at most \((n + 1)^{2^{\ell r}}\) distinct \((r, \ell)\)-partitions, where an \((r, \ell)\)-partition of an \((r, \ell)\)-graph \(G\) is a partition \((R, L)\) of \(V(G)\) such that \(G[R]\) is an \((r, 0)\)-graph and \(G[L]\) is a \((0, \ell)\)-graph. Furthermore, all these partitions can be generated in polynomial time if \(\max\{r, \ell\} \leq 2\). This result generalizes the fact that a split graph has at most \(n + 1\) split partitions \cite{10}, which was used in the algorithms of \cite{0}.

Our algorithms for \textsc{Independent} \((r, \ell)-\text{Vertex Deletion}\) (Section \(4\)) are slightly more involved, and do not explicitly use iterative compression. We rather use our algorithms for \((r, \ell)-\text{Vertex Deletion}\) for the cases \((2, 1)\) and \((2, 2)\) to obtain a set of vertices \(S\) that allows us to exploit the structure of \(G - S\). A crucial ingredient here is the FPT-algorithm of Marx et al. \cite{17} to solve the \textsc{Restricted Independent OCT} problem (see Section \(2\) for the definition).

---

**Table 1.** Summary of results for the \((r, \ell)-\text{Vertex Deletion}\) problem. Our results correspond to gray cells.

| \(p\) | \(r\)-NP-c | \(p\)-NP-c | \(p\)-NP-c | \(p\)-NP-c |
|---|---|---|---|---|
| 3 | \(1\) | \(1\) | \(1\) | \(1\) |
| 2 | OCT | 3.31k \(1\) | \(1\) | Thm \(3\) | Thm \(4\) |
| 1 | VC | 1.27k \(3\) | \(0\) | Thm \(2\) | \(1\) |
| 0 | P | \(1\) | \(1\) | \(1\) | \(1\) |

**Table 2.** Results for \textsc{Independent} \((r, \ell)-\text{Vertex Deletion}\). Our results correspond to gray cells.

| \(p\) | \(r\)-NP-c | \(p\)-NP-c | \(p\)-NP-c | \(p\)-NP-c | \(p\)-NP-c |
|---|---|---|---|---|---|
| 3 | \(1\) | \(1\) | \(1\) | \(1\) | \(1\) |
| 2 | P | 2.31k | \(3\) | Thm \(6\) | Thm \(8\) |
| 1 | VC | 1.27k | \(3\) | Thm \(6\) | Thm \(8\) |
| 0 | P | \(1\) | \(1\) | \(1\) | \(1\) |
Remarks and further research. Having completely settled the parameterized complexity of \((r, \ell)\)-Vertex Deletion and Independent \((r, \ell)\)-Vertex Deletion, a natural direction is to improve the running times of our algorithms. We did not focus in this article on optimizing the degree of the polynomial \(n^{O(1)}\) involved in our running times. Concerning the function \(f(k)\), for \((r, \ell)\)-Vertex Deletion this improvement would be possible, under ETH, only in the basis of the function \(3.31^k\) (see Theorem 5). For Independent \((r, \ell)\)-Vertex Deletion, there might be room for improvement in the function \(2^{O(\sqrt{\log k})}\) that we obtain mainly by analyzing the running time of the algorithm of Marx et al. [17] to solve Restricted Independent OCT, which was not explicit in their article.

Concerning the existence of polynomial kernels for \((r, \ell)\)-Vertex Deletion, a challenging research avenue is to apply the techniques used by Kratsch and Wahlström [12] for obtaining a randomized polynomial kernel for OCT. This improvement would be possible, under ETH, only in the basis of the function \(2^{O(\sqrt{\log k})}\) of the function \(3.31^k\) (see Theorem 5). For Independent \((r, \ell)\)-Vertex Deletion, there might be room for improvement in the function \(2^{O(\sqrt{\log k})}\) that we obtain mainly by analyzing the running time of the algorithm of Marx et al. [17] to solve Restricted Independent OCT, which was not explicit in their article.

Finally, it is worth mentioning that if the input graph is restricted to be planar, there exists a randomized subexponential algorithm for OCT [16] running in time \(O(n^{O(1)} + 2^{O(\sqrt{\log k})} n)\). As in a planar graph any clique is of size at most 4, by guessing one or two cliques and then applying this algorithm, we obtain randomized algorithms in time \(2^{O(\sqrt{\log k})} n^{O(1)}\) for \((2, 1)\)-Vertex Deletion, \((1, 2)\)-Vertex Deletion, and \((2, 2)\)-Vertex Deletion on planar graphs.

2 Preliminaries

We use standard graph-theoretic notation, and the reader is referred to [4] for any undefined term. All the graphs we consider are undirected and contain neither loops nor multiple edges. If \(S \subseteq V(G)\), we define \(G - S = G[V(G) \setminus S]\). We say that a vertex \(v\) is global to a set of vertices \(S\) if for all \(v' \in S\), \(\{v, v'\} \in E\). The complement of a graph \(G = (V, E)\) is denoted by \(\overline{G}\), that is, \(\overline{G} = (V, E')\) with \(E' = \{(x, y) \in (V \times V) \setminus E\}\). Throughout the article \(n\) denotes the number of vertices of the input graph of the problem under consideration.

It is shown in [10] that a \((1, 1)\)-graph has at most \(n + 1\) distinct \((1, 1)\)-partitions. We generalize this property in the following lemma.

Lemma 1. Let \(r\) and \(\ell\) be two fixed integers. An \((r, \ell)\)-graph on \(n\) vertices can have at most \((n+1)^{2r\ell}\) distinct \((r, \ell)\)-partitions. Moreover, if \(\max\{r, \ell\} \leq 2\), then all these partitions can be constructed in polynomial time.

Proof: Let \(G = (V, E)\) be an \((r, \ell)\)-graph, and let \((R, L)\) and \((R', L')\) be two distinct \((r, \ell)\)-partitions of \(G\). We claim that \(|R \cap L'| \leq r\ell\). Indeed, assume we have a set \(S\) of \(r\ell + 1\) vertices in \(R \cap L'\). As \(S \subseteq L'\), by the pigeonhole principle we can find a subset \(S' \subseteq S\) of size \(r + 1\) such that \(G[S']\) is a clique. As \(S' \subseteq R\), the \((r, 0)\)-graph \(G[R]\) contains a clique \(G[S']\) of size \(r + 1\), a contradiction. Symmetrically, it holds as well that \(|R' \cap L| \leq r\ell\). That is, if we fix an \((r, \ell)\)-partition \((R, L)\) of \(G\), any other \((r, \ell)\)-partition \((R', L')\) differs from \((R, L)\) by a set of at most \(2r\ell\) vertices, yielding the desired bound.
In order to generate all the partitions, first fix an \((r, \ell)\)-partition \((R, L)\) of \(G\). If \(\max\{r, \ell\} \leq 2\), then we are able to construct such an \((r, \ell)\)-partition of \(G\) in polynomial time \(^1\). We want to construct, thanks to \((R, L)\), a collection \(S\) containing all the \((r, \ell)\)-partitions of \(G\). For each \(R_g \subseteq R\) and \(L_g \subseteq L\), both of size at most \(r\ell\), we add \((L_g \cup R \setminus R_g, R_g \cup L \setminus L_g)\) to \(S\). Note that \(S\) is of size bounded by \((n + 1)^{2r\ell}\). Moreover, \(S\) contains all the possible \((r, \ell)\)-partitions of \(G\) because for any \((r, \ell)\)-partition \((R', L')\), we have \(|R \cap L'| \leq r\ell\) and \(|R' \cap L'| \leq r\ell\). Finally, we discard the elements of \(S\) that do not define an \((r, \ell)\)-partition. \(\square\)

For our algorithms we need the following restricted versions of OCT.

| **Restricted Odd Cycle Transversal (Restricted OCT)** |
|-------------------------------------------------------|
| **Input:** A graph \(G = (V, E)\), a set \(D \subseteq V\), and an integer \(k\). |
| **Parameter:** \(k\). |
| **Output:** A set \(S \subseteq D\) of size at most \(k\) such that \(G - S\) is bipartite or a correct report that such a set does not exist. |

| **Restricted Independent Odd Cycle Transversal** |
|--------------------------------------------------|
| **Input:** A graph \(G = (V, E)\), a set \(D \subseteq V\), and an integer \(k\). |
| **Parameter:** \(k\). |
| **Output:** An independent set \(S \subseteq D\) of size at most \(k\) such that \(G - S\) is bipartite or a correct report that such a set does not exist. |

**Lemma 2.** Restricted OCT can be solved in time \(2.31^k \cdot n^{O(1)}\).

*Proof:* The algorithm from Lokshtanov et al. \(^{15}\) solves OCT in time \(2.31^k \cdot n^{O(1)}\). For our lemma, we use this algorithm on a modified input. Let \(G = (V, E)\) be a graph, \(D \subseteq V\), and \(k\) an integer. We want to solve Restricted OCT on \((G, D, k)\). Let \(G' = (V', E')\), where \(V' = D \cup \{v_i : v \in V \setminus D, i \in \{0, \ldots, k\}\}\) and \(E' = (E \cap D \times D) \cup \{(v_i, w) : v \in V \setminus D, i \in \{0, \ldots, k\}, w \in D, \{v, w\} \in E\} \cup \{(v_i, w_j) : v, w \in V \setminus D, i, j \in \{0, \ldots, k\}, \{v, w\} \in E\}\). That is, for each vertex \(v\) not in \(D\), we make \(k + 1\) copies of \(v\) with the same neighborhood as \(v\), making its choice for the solution impossible. Then we solve Odd Cycle Transversal on \((G', k)\), giving us a solution of Restricted OCT on \((G, D, k)\). \(\square\)

By looking carefully the proof of [17] Theorem 4.3, we have the following theorem. We will analyze the running time of the algorithm in Subsection 4.3.

**Theorem 1 (Marx et al. [17]).** Restricted Independent OCT is FPT.

We will also need to deal with the Independent Vertex Cover problem, which given a graph \(G\) and an integer \(k\), consists in deciding whether \(G\) contains a set \(S \subseteq V(G)\) of size at most \(k\) that is both a vertex cover of \(G\) and an independent set. Theorem 6 in Section 4 proves that, in particular, Independent Vertex Cover can be solved in polynomial time on any graph, but for proving it we will first need the fact that the problem is easy on bipartite graphs.

**Lemma 3.** Independent Vertex Cover can be solved in linear time on bipartite graphs.

\(^1\) Due to space limitations, the proofs of the results marked with ‘[x]’ can be found in the appendices.
3 (r, ℓ)-Vertex Deletion

As we will use the technique of iterative compression, we need to define and solve the disjoint version of the (r, ℓ)-Vertex Deletion problem. Indeed, if we solve the disjoint version, we just need to apply Corollary 2 in Appendix A to obtain a single-exponential FPT-algorithm for (r, ℓ)-Vertex Deletion.

**Theorem 2.** Disjoint (2, 1)-Vertex Deletion can be solved in time $2.31^k \cdot n^{O(1)}$, and therefore (2, 1)-Vertex Deletion can be solved in time $3.31^k \cdot n^{O(1)}$.

**Proof:** Let $G = (V, E)$ be a graph, let $k$ be an integer, and let $S \subseteq V$ be a set of size at most $k + 1$ such that $G - S$ is a (2, 1)-graph. We want to find a set $S' \subseteq V \setminus S$ such that $G - S'$ is a (2, 1)-graph and $|S'| \leq k$, if it exists. As the property of being a (2, 1)-graph is hereditary, we can assume that $G[S]$ is a (2, 1)-graph. If it is not the case, we clearly have no solution to our problem and we stop. We fix a (2, 1)-partition $(R_0, L_0)$ of the graph $G[S]$. We also fix a (2, 1)-partition $(R_1, L_1)$ of the graph $G - S$. By Lemma 1, we know that there are at most $O(k^2 \cdot n^4)$ choices for $R_0$, $L_0$, $R_1$, and $L_1$. For each choice, we look for a solution $S' = R' \cup L'$ of size at most $k$ such that $R' \subseteq R_1$ and $L' \subseteq L_1$. A representation of this selection is depicted in Fig. 1.

![Fig. 1. An (r, ℓ)-partition of G[S] and G - S to solve Disjoint (r, ℓ)-Vertex Deletion.](image)

We define $L'$ as a smallest subset of $L_1$ such that $G[L_0 \cup (L_1 \setminus L')]$ is a clique. This set corresponds to the set of all vertices of $L_1$ not global to $L_0$. We now look for a set $R' \subseteq R_1$ of size at most $k - |L'|$ such that $G[R_0 \cup R_1 \setminus R']$ is bipartite. We find it by applying Restricted OCT to $(G[R_0 \cup R_1], R_1, k - |L'|)$. If for some guess of $R_0$, $R_1$, $L_0$, and $L_1$, the algorithm returns a solution $R'$, then we output $S' = R' \cup L'$ as our solution. Otherwise we safely return that there is no solution. Indeed, note that if $(R, L)$ is a (2, 1)-partition of $G - S'$, then $(R \cap S, L \cap S)$ is a (2, 1)-partition of $G[S]$. Moreover $(R \cap (V \setminus S), L \cap (V \setminus S))$ is a (2, 1)-partition of $G - (S \cup S')$, and since being a (2, 1)-graph is hereditary,
there is a \((2,1)\)-partition of \(G - S\) whose restriction to \(G - (S \cup S')\) is exactly \((R \cap (V \setminus S), L \cap (V \setminus S))\).

Let us now discuss about the running time of this algorithm. For each of the \(O(k^4 \cdot n^4)\) guesses, we find \(L'\) in polynomial time, and then we find \(R'\) by applying Restricted OCT. By Lemma 2, the claimed running time follows. □

The following theorem is an immediate consequence of Theorem 2.

**Theorem 3.** \((1,2)\)-Vertex Deletion can be solved in time \(3.31^k \cdot n^{O(1)}\).

**Proof:** Let \(G = (V,E)\) be a graph, \(k\) be an integer, and let \(S \subseteq V\). We just need to note that \(S\) is a solution of \((1,2)\)-Vertex Deletion on \((G,k)\) if and only if \(S\) is a solution of \((2,1)\)-Vertex Deletion on \((\bar{G},k)\).

We now deal with the \((2,2)\)-Vertex Deletion problem.

**Theorem 4.** Disjoint \((2,2)\)-Vertex Deletion can be solved in time \(2.31^k \cdot n^{O(1)}\), and therefore \((2,2)\)-Vertex Deletion can be solved in time \(3.31^k \cdot n^{O(1)}\).

**Proof:** The proof follows closely that of Theorem 2. Let \(G\) be a graph, let \(k\) be an integer, and let \(S \subseteq V\) be a set of size at most \(k + 1\) such that \(G - S\) is a \((2,2)\)-graph. We want to find \(S' \subseteq V \setminus S\) such that \(G - S'\) is a \((2,2)\)-graph. As the property of being a \((2,2)\)-graph is hereditary, we can assume that \(G[S]\) is a \((2,2)\)-graph. If it is not the case, we clearly have a \(\text{No}\)-instance and we stop. We fix a \((2,2)\)-partition \((R_0,L_0)\) of the graph \(G[S]\). We also fix a \((2,2)\)-partition \((R_1,L_1)\) of the graph \(G - S\). By Lemma 1 we know that there are at most \(O(k^8 \cdot n^8)\) choices for \(R_0, L_0, R_1\), and \(L_1\). For each choice, we look for a solution \(S' = R' \cup L'\) of size at most \(k\) such that \(R' \subseteq R_1\) and \(L' \subseteq L_1\). A representation of this selection is depicted in Fig. 4. We define \(L'\) as the smallest subset of \(L_1\) such that \(G[L_0 \cup (L_1 \setminus L')]\) is a \((0,2)\)-graph. In order to find it, we apply \(k + 1\) times the algorithm for Restricted OCT from Theorem 1 to \((\bar{G}[L_0 \cup L_1], L_1, i)\) for \(i\) from 0 to \(k\). If the algorithm does not return a solution with input \((\bar{G}[L_0 \cup L_1], L_1, k)\) then the choice of \(R_0, R_1, L_0\), and \(L_1\) is wrong, and we move to the next choice. Otherwise, let \(i_0\) be the smallest value of \(i\) for which the algorithm returns a solution, and let \(L'\) be this solution. We now look for a set \(R'\) of size at most \(k - i_0\). We find it by applying Restricted OCT to \((G[R_0 \cup R_1], R_1, k - i_0)\). If for some guess of \(R_0, R_1, L_0\), and \(L_1\), the algorithm returns a solution \(R'\), then we output \(S' = R' \cup L'\) as our solution. Otherwise we return that there is no solution.

Let us now argue about the running time. For each of the \(O(k^8 \cdot n^8)\) guesses, we find \(L'\) by applying Restricted OCT \(k + 1\) times, and then we find \(R'\) by applying Restricted OCT. By Lemma 2, the claimed running time follows. □

We conclude this section by proving easily that the running times of the previous theorems are asymptotically best possible in terms of \(k\) under ETH.

**Theorem 5.** [⋆] There is no algorithm running in time \(2^{\omega(k)} \cdot n^{O(1)}\) for solving \((2,1)\)-Vertex Deletion, \((1,2)\)-Vertex Deletion, or \((2,2)\)-Vertex Deletion unless the ETH fails.
4 Independent \((r, \ell)\)-Vertex Deletion

In this section we deal with Independent \((r, \ell)\)-VERTEX DELETION. We first provide in Subsection 4.1 a complexity dichotomy for the problem. In Subsection 4.2 we present FPT-algorithms for the cases \((2, 1)\) and \((2, 2)\). For the sake of presentation, we postpone the running time analysis of these algorithms to Subsection 4.3. As we will see, these running times strongly depend on the running time required by the algorithm of Marx et al. [17] to solve the case \((2, 0)\), that is, Independent OCT, whose bottleneck is to solve Independent Mincut.

4.1 Easy and hard cases

We first deal with the polynomially-solvable cases in Theorem 6 and then we present an \textsc{NP}-hardness reduction for the other cases in Theorem 7.

**Theorem 6.** Let \(r \in \{0, 1\}\) and \(\ell \in \{0, 1, 2\}\) be fixed integers. The Independent \((r, \ell)\)-Vertex Deletion problem can be solved in polynomial time.

**Proof:** Let \(\ell \in \{0, 1, 2\}\), let \(G = (V, E)\) be a graph, and let \(k\) be an integer. We first deal with the case \(r = 0\). We can check in polynomial time whether \(G\) is a \((1, \ell)\)-graph [1]. If it is not, then Independent \((0, \ell)\)-Vertex Deletion on \((G, k)\) has no solution. So assume that \(G\) is a \((1, \ell)\)-graph. By Lemma 1 we know that there are \(O(n^{2\ell})\) \((1, \ell)\)-partitions of \(G\). If at least one of the \((1, \ell)\)-partitions \((R, L)\) is such that \(|R| \leq k\) then we return \(R\). Otherwise, we return that no solution exists.

We now deal with the case \(r = 1\). We can check in polynomial time whether \(G\) is a \((2, \ell)\)-graph [1]. If it is not, then Independent \((1, \ell)\)-Vertex Deletion on \((G, k)\) has no solution. So assume that \(G\) is a \((2, \ell)\)-graph. By Lemma 1 we know that there are \(O(n^{4\ell})\) \((2, \ell)\)-partitions of \(G\). We guess a \((2, \ell)\)-partition \((R, L)\) of \(G\), and we want to partition \(R\) into two independent sets \(R_1\) and \(R_2\) such that \(|R_2| \leq k\). If Independent Vertex Cover on \((G[R], k)\) has a solution \(S\), then \(R_1 = R \setminus S\) and \(R_2 = S\) is the partition we want, and we return \(S\). Note that by Lemma 3 Independent Vertex Cover can be solved in linear time on the bipartite graph \(G[R]\). If Independent Vertex Cover does not return any solution over all the guesses of \((R, L)\), then we return that our problem has no solution. \(\square\)

**Theorem 7.** [*] Let \(\ell \in \{0, 1, 2\}\) be a fixed integer. The Independent \((2, \ell)\)-Vertex Deletion problem is \textsc{NP}-hard.

4.2 FPT-algorithms

We deal with the cases \((2, 1)\) and \((2, 2)\) in Theorem 8 and Theorem 9 respectively.

**Theorem 8.** Independent \((2, 1)\)-Vertex Deletion is FPT.

**Proof:** Let \(G = (V, E)\) be a graph and let \(k\) be an integer. Let \(S\) be a solution of \((2, 1)\)-Vertex Deletion on \((G, k)\). Theorem 2 gives us in time FPT such a set \(S\), or a report that no such a set exists. If there is no solution for \((2, 1)\)-Vertex Deletion on \((G, k)\) then we return that. \(\square\)
Fig. 2. An \((r, \ell)\)-partition of \(G[S \setminus I]\) and \(G - S\) to solve INDEPENDENT \((r, \ell)\)-VERTEX DELETION.

DELETION, then INDEPENDENT \((2, 1)\)-VERTEX DELETION has no solution either. So we assume that such a set \(S\) exists. Using \(S\), we proceed to construct a solution \(S'\) of our problem as follows. We first guess an independent subset \(I\) of \(S\). (Note that there are at most \(2^k\) such subsets.) We want to construct the solution \(S'\) such that \(I \subseteq S'\) and \(S' \cap S = I\). If \(G[S \setminus I]\) is not a \((2, 1)\)-graph then our choice of \(I\) is wrong. So assume \(G[S \setminus I]\) is a \((2, 1)\)-graph, and let \((R_0, L_0)\) be a \((2, 1)\)-partition of \(G[S \setminus I]\). Let \((R_1, L_1)\) be a \((2, 1)\)-partition of \(G - S\). By Lemma 1 we know that there are at most \(O(k^4 \cdot n^4)\) choices for \(R_0, L_0, R_1,\) and \(L_1\). We want to find \(R' \subseteq R_1\) and \(L' \subseteq L_1\) such that \(S' = I \cup L' \cup R'\). A representation of this selection is depicted in Fig. 2 As we want the solution to induce an independent set, at most one element of \(L_1\) is in \(S'\), that is, \(|L'| \leq 1\). If there is at most one element of \(L_1\) that is not global to \(L_0\) and not in the neighborhood of \(I\), then we have our set \(L'\) with at most one element. Otherwise, our choices were wrong. We now have to find \(R'\) of size at most \(k - |I| - |L'|\). For this, we apply the algorithm for Restricted INDEPENDENT OCT of Theorem 4 on \((G[R_0 \cup R_1], D, k - |I| - |L'|)\) with \(D = \{x \in R_1 : \forall y \in I \cup L', (x, y) \notin E\}\). If it returns a solution \(R'\) then we can output the solution \(S' = I \cup L' \cup R'\). Note that \(S'\) is an independent set by construction. If it does not return a solution for any of the guesses of \(I, R_0, L_0, R_1,\) and \(L_1\), then we return that there is no solution of INDEPENDENT \((2, 1)\)-VERTEX DELETION.

\[\blacksquare\]

**Theorem 9.** INDEPENDENT \((2, 2)\)-VERTEX DELETION is FPT.

**Proof:** The proof follows closely that of Theorem 8. Let \(G = (V, E)\) be a graph and let \(k\) be an integer. Let \(S\) be a solution of the \((2, 2)\)-VERTEX DELETION problem on \((G, k)\). Theorem 4 gives us in time FPT such a set \(S\), or a report that such a set does not exist. If there is no solution for \((2, 2)\)-VERTEX DELETION, then INDEPENDENT \((2, 2)\)-VERTEX DELETION has no solution either. So we can assume that such a set \(S\) exists. Using \(S\), we proceed to construct a solution \(S'\) of our problem as follows. We first guess an independent subset \(I\) of \(S\). We want to construct the solution \(S'\) such that \(I \subseteq S'\) and \(S' \cap S = I\). If \(G[S \setminus I]\) is not a \((2, 2)\)-graph then our choice of \(I\) is wrong. So assume \(G[S \setminus I]\) is a \((2, 2)\)-graph, and let \((R_0, L_0)\) be a \((2, 2)\)-partition of \(G[S \setminus I]\). Let \((R_1, L_1)\) be a \((2, 2)\)-partition
of $G - S$. By Lemma 1, we know that there are at most $O(k^8 \cdot n^8)$ choices for $R_0$, $L_0$, $R_1$, and $L_1$. We want to find $R' \subseteq R_1$ and $L' \subseteq L_1$ such that $S' = I \cup L' \cup R'$. A representation of this selection is depicted again in Fig. 2. As we want the solution to induce an independent set, at most two elements of $L_1$ are in $S'$, that is, $|L'| \leq 2$. We guess these at most two vertices that define $L'$ such that $L' \cup I$ is an independent set and such that $G[(L_0 \cup L_1) - L']$ is a $(0, 2)$-graph. If it is not the case then our choice is wrong. We now have to find $R'$ of size at most $k - |I| - |L'|$. For this, we apply Restricted Independent OCT on $(G[R_0 \cup R_1], D, k - |I| - |L'|)$ with $D = \{x \in R_1 : \forall y \in I \cup L', \{x, y\} \notin E\}$. If it returns a solution $R'$ then we can output the solution $S' = I \cup L' \cup R'$. If it does not return a solution for any of the guesses of $I$, $R_0$, $L_0$, $R_1$, and $L_1$, then we return that there is no solution of Independent $(2, 2)$-Vertex Deletion. 

4.3 Analysis of the running time

In this subsection we provide an upper bound on the running times of the FPT-algorithms for Independent $(2, 1)$-Vertex Deletion and Independent $(2, 2)$-Vertex Deletion given by Theorem 8 and Theorem 9 respectively. Note that in these algorithms, the only non-explicit running time is the one of the algorithm for Restricted Independent OCT given by Theorem 1. To obtain this upper bound, we will go through the main ideas of the algorithm of Marx et al. [17] for Independent OCT, and then by using the same tools used in the proof of Lemma 2 we will obtain the same upper bound for the restricted version of Independent OCT.

We need to define the following problem, where an $s-t$ cut in a graph $G$ is a set of vertices $C$ such that $s$ is not connected to $t$ in the graph $G - C$.

### Independent Mincut

**Input:** A graph $G = (V, E)$, an integer $k$, and two vertices $s, t \in V$.

**Parameter:** $k$.

**Output:** An $s-t$ cut $C \subseteq V$ such that $|C| \leq k$ and $C$ is an independent set, or a correct report that such a set does not exist.

We provide here a sketch of proof of the following simple lemma. (For completeness, the definition of treewidth can be found in Appendix [E].)

**Lemma 4.** Independent Mincut can be solved in time $3^{\text{tw}} \cdot n^{O(1)}$, where tw stands for the treewidth of the input graph.

**Proof:** (Sketch) For each bag $B$ of the tree-decomposition, we store all quadruples $(S, T, D, \ell)$ such that we have already found a cut $C'$ of size at most $\ell$ in the explored graph such that $B \cap C' = D$, such that there is no edge between $S$ and $T$, and such that $s \in B$ if and only if $s \in S$ and $t \in B$ if and only if $t \in T$. There are at most $3^{\text{tw}} \cdot k$ such quadruples, and so the lemma follows. 

We also need the following result, where the key idea is to obtain an equivalent graph whose treewidth is bounded by a function of $k$. 

---

[1] Marx et al. [17]

[2] Marx et al. [17]

[3] Marx et al. [17]
Theorem 10 (Marx et al. [17]). Let $G = (V, E)$ be a graph, let $S \subseteq V(G)$, and let $k$ be an integer. Let $C$ be the set of all vertices of $G$ participating in a minimal $s - t$ cut of size at most $k$ for some $s, t \in S$. Then there is an algorithm running in time $2^{O(k^2)} \cdot n^{O(1)}$ that computes a graph $G^*$ having the following properties:

- $C \cup S \subseteq V(G^*)$,
- For every $s, t \in S$, a set $K \subseteq V(G^*)$ with $|K| \leq k$ is a minimal $s - t$ cut of $G^*$ if and only if $G^*$ has a solution such that $s$ is in $Y$ and $t$ is in $X$.
- The treewidth of $G^*$ is at most $O(k^2)$, and
- For any $K \subseteq C$, $G^*[K]$ is isomorphic to $G[K]$.

Lemma 5. INDEPENDENT MINCUT can be solved in time $2^{O(k^2)} \cdot n^{O(1)}$.

Proof: Let $G = (V, E)$ be a graph and $k$ be an integer. Let $G^*$ be the graph satisfying the requirements of Theorem 10 for $S = \{s, t\}$. Following the proof of [17] Theorem 3.1, it follows that $(G, s, t, k)$ has a solution of INDEPENDENT MINCUT if and only if $(G^*, s, t, k)$ has one. We can now apply Lemma 4 and solve INDEPENDENT MINCUT on $(G^*, s, t, k)$ in time $3^{\text{tw}(G^*)} \cdot n^{O(1)}$. By Theorem 10, $\text{tw}(G^*) = 2^{O(k^2)}$ and the lemma follows. \qed

Note that the restricted version of INDEPENDENT MINCUT can be solved within the same running time by making enough copies of each “undesired” vertex, as in the proof of Lemma 2.

Theorem 11. INDEPENDENT OCT can be solved in time $2^{O(k^2)} \cdot n^{O(1)}$.

Proof: Let $G = (V, E)$ be a graph and let $X$ be a solution of OCT on $(G, k)$, which we can assume to exist. Let $(S_1, S_2)$ be a partition of $G - X$ into two independent sets. We define an auxiliary graph $G' = (V', E')$ as defined in [19]. So we have $V' = (V \setminus X) \cup \{x_1, x_2 : x \in X\}$ and $E' = \{\{v, w\} \in E : v, w \in V \setminus X\} \cup \{\{y, x_{i-1}\} : y \in S_i, x \in X, i \in \{1, 2\}, \{x, y\} \in E\} \cup \{\{x_1, y_2\} : x, y \in X, \{x, y\} \in E\}$. Given $Y \subseteq X$, we say that a partition of $Y' = \{y_1, y_2 : y \in Y\}$ into two sets $(Y_A, Y_B)$ is valid if for all $y \in Y$, exactly one of $y_1, y_2$ is in $Y_A$. We let $S = S_1 \cup S_2$.

To continue, we need the next reformulation of [19] Lemma 1 and its proof.

Claim 1. [x] There is an independent odd cycle transversal $Z$ of size at most $k$ in $G$ if and only if there exists $Y \subseteq X$ and a valid partition $(Y_A, Y_B)$ of $Y'$ such that there is an independent mincut $C \subseteq S$ that separates $Y_A$ from $Y_B$ in $G'$ and such that $Z = C \cup (X \setminus Y)$ is an independent set of size at most $k$ in $G$.

We now apply INDEPENDENT MINCUT for all $Y \subseteq X$ and all valid partitions $(Y_A, Y_B)$ of $Y'$. Note that we need to apply a restricted version of INDEPENDENT MINCUT because we do not want the neighborhood of $X \setminus Y$ to be in the solution. By Claim 1 if we obtain a solution, then we have found our independent odd cycle transversal, and otherwise we can safely return that such a set does not exist. The claimed running time follows by Lemma 5. \qed

By using the same argument of Lemma 2, we obtain the following corollary.
Corollary 1. Restricted Independent OCT can be solved in time $2^{O(k^2)} \cdot n^{O(1)}$, and therefore Independent $(r, \ell)$-Vertex Deletion can also be solved in time $2^{O(k^2)} \cdot n^{O(1)}$ for $r = 2$ and $\ell \in \{0, 1, 2\}$.

Note that the previous results would be automatically improved if one can find a faster algorithm for Independent Mincut.

References

1. A. Brandstädt. Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics*, 152(1-3):47–54, 1996.
2. L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.
3. J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.
4. R. Diestel. *Graph Theory*. Springer-Verlag, Berlin, 3rd edition, 2005.
5. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
6. T. Feder, P. Hell, S. Klein, and R. Motwani. List partitions. *SIAM Journal on Discrete Mathematics*, 16(3):449–478, 2003.
7. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
8. S. Foldes and P. Hammer. Split graphs. *Congressus Numerantium*, 19:311–315, 1977.
9. E. Ghosh, S. Kolay, M. Kumar, P. Misra, F. Panolan, A. Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015.
10. M. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier, 2004.
11. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
12. S. Kratsch and M. Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms*, 10(4):20, 2014.
13. R. Krithika and N. S. Narayanaswamy. Parameterized algorithms for $(r, l)$-partition. *Journal of Graph Algorithms and Applications*, 17(2):129–146, 2013.
14. J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.
15. D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15, 2014.
16. D. Lokshtanov, S. Saurabh, and M. Wahlström. Subexponential parameterized odd cycle transversal on planar graphs. In *Proc. of FSTTCS*, volume 18 of *LIPIcs*, pages 424–434, 2012.
17. D. Marx, B. O’Sullivan, and I. Razgon. Treewidth Reduction for Constrained Separation and Bipartition Problems. In *Proc. of STACS*, volume 5 of *LIPIcs*, pages 561–572, 2010.
18. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
19. B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.
A Well-known properties of iterative compression

As mentioned in the introduction, iterative compression has been successfully used to obtain efficient algorithms for a number of parameterized problems [9,12,19]. In a nutshell, the main idea of this technique is to reduce in FPT-time a problem to solving a compressed (usually also called disjoint) version of it where we assume that we are given a solution of size almost as small as the desired one, and that allows us to exploit the structure of the graph in order to obtain the actual solution. This technique usually applies to hereditary properties.

A graph property $Q$ is hereditary if any subgraph of a graph that satisfies $Q$ also satisfies $Q$. Let $Q$ be a hereditary graph property. We define the following two problems in order to state two general facts about the technique of iterative compression, which we use in Section 3.

**Q-Vertex Deletion**

**Input:** A graph $G = (V, E)$ and an integer $k$.

**Parameter:** $k$.

**Output:** A set $S \subseteq V$ of size at most $k$ such that $G - S$ satisfies property $Q$, or a correct report that such a set does not exist.

**Disjoint Q-Vertex Deletion**

**Input:** A graph $G = (V, E)$, an integer $k$, and a set $S \subseteq V$ of size at most $k + 1$ such that $G - S$ satisfies property $Q$.

**Parameter:** $k$.

**Output:** A set $S' \subseteq V \setminus S$ of size at most $k$ such that $G - S'$ satisfies property $Q$, or a correct report that such a set does not exist.

The following two results are well-known and commonly assumed when using iterative compression. We include the proofs here for completeness.

**Lemma 6.** If Disjoint Q-Vertex Deletion can be solved in FPT-time, then Q-Vertex Deletion can also be solved in FPT-time.

**Proof:** Let $A$ be an FPT algorithm which solves Disjoint Q-Vertex Deletion. Let $G = (V, E)$ be a graph and $k$ be an integer. We want to solve Q-Vertex Deletion on $(G, k)$. Let $v_1, \ldots, v_n$ be an arbitrary ordering of $V$. For each $i \in \{0, \ldots, n\}$, let $V_i$ denote the subset of vertices $\{v_1, \ldots, v_i\}$ and $G_i = G[V_i]$. We iterate over $i$ from 1 to $n$ as follows. At the $i$-th iteration, suppose we have a solution $S_i \subseteq V_i$ of Q-Vertex Deletion on $(G_i, k)$. At the next iteration, we can define $S_{i+1} = S_i \cup \{v_{i+1}\}$. Note that $S_{i+1}$ is a solution of Q-Vertex Deletion on $(G_{i+1}, k+1)$. If $S_{i+1}$ is of size at most $k$ then it is a solution of Q-Vertex Deletion on $(G_{i+1}, k)$. Assume that $S_{i+1}$ is of size exactly $k + 1$. We guess a subset $S$ of $S_{i+1}$ and we look for a solution $W$ of Q-Vertex Deletion on $(G_{i+1}, k)$ that does not contain any element of $S$. For it, we use algorithm $A$ on $(H, |S| - 1, S)$ with $H = G_{i+1} - (S_{i+1} \setminus S)$. If $A$ returns a solution $W$ then observe that the set $W \cup (S_{i+1} \setminus S)$ is a solution of Q-Vertex
Deletion on $(G_{i+1}, k)$. If $A$ on $(H, |S| - 1, S)$ does not return a positive answer for any of the possible guesses of $S$, then $Q$-Vertex Deletion on $(G_{i+1}, k)$ has no solution. Since the property $Q$ is hereditary, $Q$-Vertex Deletion on $(G, k)$ has no solution either, and therefore the algorithm returns that there is no solution. Thus, we obtain an algorithm solving $Q$-Vertex Deletion in FPT-time, as we wanted.

Corollary 2. If Disjoint $Q$-Vertex Deletion can be solved in time $c^k \cdot n^{O(1)}$ for some constant $c$, then $Q$-Vertex Deletion can be solved in time $(c + 1)^k \cdot n^{O(1)}$.

Proof: Let us argue about the running time of the algorithm of Lemma 6 assuming Disjoint $Q$-Vertex Deletion can be solved in time $c^k \cdot n^{O(1)}$ for some constant $c$. The time required to execute $A$ for every subset $S$ at the $i$-th iteration is $\sum_{i=0}^{k+1} \binom{k+1}{i} \cdot c^i \cdot n^{O(1)} = (c + 1)^{k+1} \cdot n^{O(1)}$. We obtain an algorithm that computes $P(Q)$ in time $(c + 1)^k \cdot n^{O(1)}$, as we wanted.

B Proof of Lemma 3

Let $G = (V, E)$ be a bipartite graph, and we proceed to construct a solution $S$ of minimum size. For each connected component of $G$, we define the unique bipartition $(B_1, B_2)$ such that $|B_1| < |B_2|$ and $B_1$ and $B_2$ are two independent sets. (If $|B_1| = |B_2|$, we arbitrarily choose $B_1$ being one of them and $B_2$ being the other one.) Note that $S$ cannot contain vertices in both $B_1$ and $B_2$, since in that case by connectivity there would exist an alternating path in $G$ with only the endvertices in $S$, and then either there is an edge between both endvertices (contradicting the fact that $S$ should be an independent set), or some edge in the path does not contain vertices in $S$ (contradicting the fact that $S$ should be a vertex cover). Thus, if $S$ is a minimum-size solution, necessarily $S \cap (B_1 \cup B_2) = B_1$. Therefore, we start with $S = \emptyset$, and for each connected component of $G$, we add each element of $B_1$ to $S$. After exploring the whole graph, if $|S| \leq k$ then we return $S$, otherwise we report that no such a set exists.

C Proof of Theorem 5

It is known that $(2, 0)$-Vertex Deletion cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$ unless the ETH fails [14][15]. We proceed to reduce $(2, 0)$-Vertex Deletion to $(2, 1)$-Vertex Deletion while do not changing the parameter and so that the size of the resulting graph is polynomial on the size of the original one. Let $G = (V, E)$ be a graph, let $k$ be an integer, and let $K_{k+3} = (V_K, E_K)$ be the complete graph on $k+3$ vertices. Let $G'$ be the disjoint union of $G$ and $K_{k+3}$. Let $S$ be a solution of $(2, 0)$-Vertex Deletion on $(G, k)$. Thus, $G - S$ is a $(2, 0)$-graph, which directly implies that $G' - S$ is a $(2, 1)$-graph. So $S$ is a solution of $(2, 1)$-Vertex Deletion on $(G', k)$. Now assume that $S$ is a solution of $(2, 1)$-Vertex Deletion on $(G', k)$. Then $G' - S$ is a $(2, 1)$-graph. As $K_{k+3} - (S \cap V_K)$ is a clique containing at least 3 vertices that is not connected to $G$, any $(2, 1)$-partition $(R, L)$ of $G' - S$ is such that $L \subseteq V_K \setminus S$. As $V_K \setminus (S \cap V_K)$ induces...
a clique, without loss of generality, we can assume that $L = V_K \setminus S$. Thus, $R = V \setminus S$. As $G'[R]$ is bipartite, $S \cap V$ is a solution of $(2, 0)$-Vertex Deletion on $(G, k)$.

A similar proof can be given for $(2, 2)$-Vertex Deletion by adding two large disjoint cliques instead of one. On the other hand, as $S$ is a solution of $(1, 2)$-Vertex Deletion on $(G, k)$ if and only if $S$ is a solution of $(2, 1)$-Vertex Deletion on $(G', k)$, the theorem is also valid for $(1, 2)$-Vertex Deletion.

D Proof of Theorem 7

We first prove that Independent $(2, 0)$-Vertex Deletion is NP-hard. We reduce from $(2, 0)$-Vertex Deletion, commonly called Odd Cycle Transversal. The problem is proved to be NP-complete in [14].

Let $G = (V, E)$ be a graph, let $k$ be an integer, and let $n = |V|$. We want to solve $(2, 0)$-Vertex Deletion on $(G, k)$. We define $G' = (V', E')$, such that $V' = V \cup \{v_i^e : v \in V, e = \{v, w\} \in E, i \in \{0, \ldots, n\}\}$ and $E' = \{v, v_i^e : v \in V, e = \{v, w\} \in E, i \in \{0, \ldots, n\}\} \cup \{v_i^e, w_i^e : v \in V, w \in V, e = \{v, w\} \in E, i \in \{0, \ldots, n\}\}$. That is, we replace each edge $e = \{v, w\}$ of $E$ by $n + 1$ paths $v, v_i^e, w_i^e, w$ of size 4, for $i \in \{0, \ldots, n\}$. Assume we have a solution $S \subseteq V$ of $(2, 0)$-Vertex Deletion on $(G, k)$. In $G'$, there is no edge between two vertices of $V$. So $S$ is also a solution of Independent $(2, 0)$-Vertex Deletion on $(G', k)$. Now, we assume that $S$ is a solution of Independent $(2, 0)$-Vertex Deletion on $(G', k)$. Then we have that $S \cap V$ is also a solution of Independent $(2, 0)$-Vertex Deletion on $G'$. Indeed, assume $v_i^e \in V'$ is in $S$ for some $e = \{v, w\} \in E$ and $i \in \{0, \ldots, n\}$. If $v \in S$ then $v_i^e$ has only one neighbor in $G' \setminus \{v\}$, so if $G - S$ is bipartite, then so is $G - (S \setminus \{v_i^e\})$ and thus $S \setminus \{v_i^e\}$ is also a solution. The same happens if $w \in S$. So assume now that $v$ and $w$ are not in $S$. Then there exists at least one index $i' \in \{0, \ldots, n\}$ such that $v_i^{e'}$ and $w_i^{e'}$ are not in $S$. It means that in the bipartite graph $G - S, v$ and $w$ are in two different independent sets. We can safely add $v_i^e$ to $G - S$ such that the graph remains bipartite by adding $v_i^e$ to the independent set containing $w$. So $S \setminus \{v_i^e\}$ is also a solution. By deleting all the vertices of the form $v_i^e$ from $S$, we obtain a set $S'$ such that $S' \subseteq V$ and $|S'| \leq k$. As we preserve the property in $G'$ that if $\{v, w\} \in E$, with $v$ and $w$ not in $S'$, then $v$ and $w$ should be in two different independent sets in $G' - S'$, we have that $S'$ is a solution of $(2, 0)$-Vertex Deletion on $G$. This concludes the proof.

We now show that Independent $(2, 1)$-Vertex Deletion is NP complete by reducing from Independent $(2, 0)$-Vertex Deletion. Let $G = (V, E)$ be a graph and let $k$ be an integer. Let $n = |V|$ and $K_{n+3} = (V_K, E_K)$ be the complete graph on $n + 3$ vertices. We can assume that $k \leq n$. Let $G'$ be the disjoint union of $G$ and $K_{n+3}$. Let $S$ be a solution of Independent $(2, 0)$-Vertex Deletion on $(G, k)$. Thus, $G - S$ is a $(2, 0)$-graph, and so $G' - S$ is a $(2, 1)$-graph. It follows that $S$ is a solution of Independent $(2, 1)$-Vertex Deletion on $(G', k)$. Now assume that $S$ is a solution of Independent $(2, 1)$-Vertex Deletion on $(G', k)$. Then $G' - S$ is a $(2, 1)$-graph. As $V_K \setminus S$ induces a clique containing at least 3 vertices that is not connected to $G$, any $(2, 1)$-partition
(R, L) of $G' - S$ is such that $L \subseteq V_K \setminus S$. As $V_K \setminus S$ induces a clique, without loss of generality we can assume that $L = V_K \setminus S$. It follows that $R = V \setminus S$.

Finally, as $G'[R]$ is bipartite, $S \cap V$ is a solution of INDEPENDENT $(2, 0)$-VERTEX DELETION on $(G, k)$.

With a similar proof, we show that INDEPENDENT $(2, 2)$-VERTEX DELETION is also NP-hard.

E  Tree-decompositions and treewidth

A tree-decomposition of width $w$ of a graph $G = (V, E)$ is a pair $(T, \sigma)$, where $T$ is a tree and $\sigma = \{B_t | B_t \subseteq V, t \in V(T)\}$ such that:

- $\bigcup_{t \in V(T)} B_t = V$,
- For every edge $\{u, v\} \in E$ there is a $t \in V(T)$ such that $\{u, v\} \subseteq B_t$,
- $B_i \cap B_k \subseteq B_j$ for all $\{i, j, k\} \subseteq V(T)$ such that $j$ lies on the path $i, \ldots, k$ in $T$, and
- $\max_{t \in V(T)} |B_t| = w + 1$.

The sets $B_t$ are called bags. The treewidth of $G$, denoted by $tw(G)$, is the smallest integer $w$ such that there is a tree-decomposition of $G$ of width $w$. An optimal tree-decomposition is a tree-decomposition of width $tw(G)$.

F  Proof of Claim I

$(\Rightarrow)$ Let $Z$ be an independent odd cycle transversal of size at most $k$ in $G$. We assume that $Z$ is of minimum size and that its removal produces two independent sets $S^Z_1$ and $S^Z_2$. Let $K = Z \cap X$, let $Z' = Z \setminus K$, and let $Y = X \setminus K$ with valid partition $(Y_A, Y_B)$ defined by $Y_A = \{y_1 : y \in Y \cap S^Z_1\} \cup \{y_2 : y \in Y \cap S^Z_2\}$ and $Y_B = \{y_2 : y \in Y \cap S^Z_1\} \cup \{y_1 : y \in Y \cap S^Z_2\}$. We claim that $Z'$ is a cutset of $G'[Y_A \cup Y_B \cup S]$. Take a minimal path $P$ from $Y_A$ to $Y_B$ in $G'[Y_A \cup Y_B \cup S] \setminus Z'$. Let $u$ and $v$ be the endpoints of $P$. By minimality of $P$, $P \cap (Y_A \cup Y_B) = \{u, v\}$. We assume without loss of generality that either $u \in Y \cap S^Z_1$ or $u \in Y \cap S^Z_2$ and $v \in Y \cap S^Z_2$. In the first case, we have that $u = y_1$ for some $y \in Y$ and $v = w_2$ for some $w \in Y$. So $P$ has odd size. But as $u, v \in S^Z_2$, then $P$ should have even size. We obtain a similar contradiction for the second case.

$(\Leftarrow)$ Under the condition of Claim I, $Z$ is an independent set, and we need to prove that it is an odd cycle transversal as well. Assume that there is an odd cycle $O$ in $G - Z$. Then by definition of $X$, $O$ intersects $X$ at least once. Let $O^0, \ldots, O^{m-1}$ be the $m$ times $O$ intersects $X$ and we define $O^m = O^0$. We have that $O^i \neq O^j$ for all $i < j < m$. For each $i \in \{0, \ldots, m - 1\}$, let $P_i$ be the path from $O^i$ to $O^{i+1}$.

As $O$ never intersects $Z$, then in $G'$ the path $P_i$ never goes from $Y_A$ to $Y_B$. It means that for each $i$ such that $P_i$ is of even size, $O^i_1$ and $O^{i+1}_1$ are in the same set $Y_A$ or $Y_B$, and for each $i$ such that $P_i$ is of odd size, $O^i_1$ and $O^{i+1}_2$ are in the same set $Y_A$ or $Y_B$. But $O$ is an odd cycle, so there is an odd number of paths $P_i$ such that $P_i$ is of odd size. We deduce that such an odd cycle $O$ cannot exist, meaning that $Z$ is an independent odd cycle transversal.