Complexity of Finding Maximum Locally Irregular Induced Subgraphs

Foivos Fioravantes ✉
Université Côte d'Azur, Inria, CNRS, I3S, Valbonne, France

Nikolaos Melissinos ✉
Université Paris-Dauphine, Université PSL, CNRS, LAMSADE, 75016, Paris, France

Theofilos Triommatis ✉
School of Electrical Engineering, Electronics and Computer Science, University of Liverpool, UK

Abstract

If a graph G is such that no two adjacent vertices of G have the same degree, we say that G is *locally irregular*. In this work we introduce and study the problem of identifying a largest induced subgraph of a given graph G that is locally irregular. Equivalently, given a graph G, find a subset S of $V(G)$ with minimum order, such that by deleting the vertices of S from G results in a locally irregular graph; we denote with $\mathrm{I}(G)$ the order of such a set S. We first examine some easy graph families, namely paths, cycles, trees, complete bipartite and complete graphs. However, we show that the decision version of the introduced problem is $\mathcal{NP}$-Complete, even for restricted families of graphs, such as subcubic planar bipartite, or cubic bipartite graphs. We then show that we can not even approximate an optimal solution within a ratio of $\mathcal{O}(n^{1-\frac{1}{k}})$, where $k \geq 1$ and n is the order the graph, unless $\mathcal{P}$=$\mathcal{NP}$, even when the input graph is bipartite.

Then, looking for more positive results, we turn our attention towards computing $\mathrm{I}(G)$ through the lens of parameterised complexity. In particular, we provide two algorithms that compute $\mathrm{I}(G)$, each one considering different parameters. The first one considers the size of the solution k and the maximum degree Δ of G with running time $(2\Delta)^k n^{\mathcal{O}(1)}$, while the second one considers the treewidth tw and Δ of G, and has running time $\Delta^{2tw} n^{\mathcal{O}(1)}$. Therefore, we show that the problem is FPT by both k and tw if the graph has bounded maximum degree Δ. Since these algorithms are not FPT for graphs with unbounded maximum degree (unless we consider $\Delta + k$ or $\Delta + tw$ as the parameter), it is natural to wonder if there exists an algorithm that does not include additional parameters (other than k or tw) in its dependency.

We answer negatively, to this question, by showing that our algorithms are essentially optimal. In particular, we prove that there is no algorithm that computes $\mathrm{I}(G)$ with dependence $f(k)n^{o(k)}$ or $f(tw)n^{o(tw)}$, unless the ETH fails.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Approximation algorithms

Keywords and phrases Locally irregular, largest induced subgraph, FPT, treewidth, W-hardness, approximability

Digital Object Identifier 10.4230/LIPIcs.SWAT.2022.24

Related Version *Full Version*: `https://hal.archives-ouvertes.fr/hal-03358273`

Funding *Foivos Fioravantes*: Supported by the French government through the UCA JEDI (ANR-15-IDEX-01) and the EUR DS4H (ANR-17-EURE-004) Investments in the Future projects.
Theofilos Triommatis: Supported by EP/S023445/1 EPSRC CDT in Distributed Algorithms, University of Liverpool.

1 Introduction

A graph G is said to be *locally irregular*, if every two adjacent vertices of G have different degrees. In this paper, we introduce and study the problem of finding a largest locally irregular induced subgraph of a given graph. This problem is equivalent to identifying what is the minimum number of vertices that must be deleted from G, so that what remains is a locally irregular graph.

Locally irregular graphs. The notion of locally irregular graphs was first introduced in [6]. The most interesting aspect of locally irregular graphs, comes from their connection to the so-called 1-2-3 Conjecture, proposed in [22]. Formally, the 1-2-3 Conjecture states that for almost every graph, we should be able to place weights from $\{1, 2, 3\}$ on the edges of that graph, so that the colouring, that assigns a colour to each vertex equal to the sum of the weights on its adjacent edges, is a proper vertex-colouring of the graph.

As we said earlier, the 1-2-3 Conjecture seems to have some very interesting links to locally irregular graphs. An obvious connection is that this conjecture holds for locally irregular graphs. Indeed, placing weight equal to 1 to all the edges of a locally irregular graph, suffices to produce a proper vertex-colouring, as each vertex receives a colour equal to its degree. Furthermore, there have been some steps towards proving that conjecture, which involve edge-decomposing a graph into a constant number of locally irregular subgraphs, i.e., given G, find an edge-colouring of G using a constant number of colours, such that each colour induces a locally irregular subgraph of G. This is the main motivation behind [6], and it seems to remain interesting enough to attract more attention [8, 25, 30].

Note that the class of locally irregular graphs can be seen as an antonym to that of *regular* graphs, i. e., graphs such that all of their vertices have the same degree. It is important to state here that there exist several alternative such notions. This is mainly due to the very well known fact that there are no non-trivial *irregular* graphs, i. e., graphs that do not contain two vertices (not necessarily adjacent) with the same degree (see [12]). Thus, the literature has plenty of slightly different definitions of irregularity (see for example [2, 12, 13, 20, 29]). One way to deal with the nonexistence of irregular graphs, is to define a notion of *local* irregularity. Intuitively, instead of demanding for all vertices of a graph to have different degrees, we are now considering each vertex v separately, and request that the vertices "around" v to verify some properties of irregularity. For example, the authors of [3] study graphs G such that for every vertex v of G, no two neighbours of v have the same degree. For an overview of other interesting notions of irregularity (local or otherwise), we refer the reader to [4].

Largest induced subgraph verifying specific properties. The problem we introduce belongs in a more general and well studied family of problems, which is about identifying a largest induced subgraph of a given graph that verifies a specific property Π. That is, given a graph $G = (V, E)$ and an integer k, is there a set $V' \subseteq V$ such that $|V'| \leq k$ and $G[V \setminus V']$ has the specified property Π? In our case, the property Π is "the induced subgraph is locally irregular". This generalised problem is indeed classic in graph theory, and it is known as the Induced Subgraph with Property Π (ISPΠ for short) problem in [21]. Unfortunately, it was shown in [24], that ISPΠ is a hard problem for any property Π that is *hereditary*, i. e., all induced subgraphs of G verify Π if G itself verifies that property.

However, the ISPΠ problem remains interesting (one could say that it actually becomes more interesting) even if the property Π is not hereditary. Recently, the authors of [7] studied the problem for Π being "all vertices of the induced subgraph have odd degree", which clearly is not a hereditary property. Nevertheless, they showed that this is an $\mathcal{NP}$-hard problem, and they gave an FPT algorithm that solves the problem when parameterised by the rank-width. Also, the authors of [1, 5, 28] studied the ISPΠ problem, where Π is the rather natural property "the induced subgraph is d-regular", where d is an integer given in the input (recall that a graph is said to be *d-regular* if all of its vertices have the same degree d). In particular, in [5] it is shown that finding a largest (connected) induced subgraph that is d-regular, is $\mathcal{NP}$-hard to approximate, even when restricted on bipartite or planar graphs. The authors of [5] also provide a linear-time algorithm to solve this problem for graphs with bounded treewidth. In contrast, the authors of [1] take a more practical approach, as they focus on solving the problem for the particular values of $d = 1$ and $d = 2$, by using bounds from quadratic programming, Lagrangian relaxation and integer programming.

It is quite clear that, in some sense, the property that interests us lies on the opposite side of the one studied in [1, 5, 28]. However, both properties, "the induced subgraph is regular" and "the induced subgraph is locally irregular" are not hereditary. This means that we do not get an $\mathcal{NP}$-hardness result directly from [24]. Furthermore, the ISPΠ problem always admits an FPT algorithm, when parameterised by the size of the solution, if Π is a hereditary property (proven in [11, 23]), but for a non-hereditary one, this is not always true. Indeed in [28], the authors proved that when considering Π as "the induced subgraph is regular", the ISPΠ problem is W[1]-hard when parameterised by the size of the solution. That is, there should be no $f(k)n^c$ time algorithm for this problem, where c is a constant. For such problems, it is also interesting to see if there exists any algorithm with running time $n^{o(k)}$ or $f(k)n^{o(k)}$. The authors of [14, 15, 16] provide techniques that can be used to strongly indicate the non-existence of such algorithms, by applying them on a variety of W[1]-hard and W[2]-hard problems, such as the Independent Set and the Dominating Set, parameterised by the size of their solutions. Usually these lower bounds are shown under the assumption of a weaker version of the Exponential Time Hypothesis, which states that SAT can not be solved in time $2^{o(n+m)}$.

Our contribution. We begin in Section 2 by providing the basic notations and definitions that are going to be used throughout this paper. In Section 3, we deal with the complexity of the introduced problem. In particular, we show that the problem belongs in $\mathcal{P}$ if the input graph is a path, cycle, tree, complete bipartite or complete graph. We then prove that finding the maximum induced locally irregular subgraph of a given graph G is $\mathcal{NP}$-hard, even if G is restricted to being a subcubic planar bipartite, or a cubic bipartite graph.

As the introduced problem seems to be computationally hard even for restricted families of graphs, we then investigate its approximability. Unfortunately, we prove in Section 4 that for any bipartite graph G of order n and $k \geq 1$, there can be no polynomial time algorithm that finds an approximation of $\mathrm{I}(G)$ within ratio $\mathcal{O}(n^{1-\frac{1}{k}})$, unless $\mathcal{P}$=$\mathcal{NP}$. Nevertheless, we do provide a (simple) d-approximation algorithm for d-regular bipartite graphs.

We then decide to look into its parameterised complexity. In Section 5, we present two algorithms that compute $\mathrm{I}(G)$, each one considering different parameters. The first considers the size of the solution k and the maximum degree Δ of G, and and has running time $(2\Delta)^k n^{\mathcal{O}(1)}$, while the second considers the treewidth tw and Δ of G, and has running time $\Delta^{2tw} n^{\mathcal{O}(1)}$. Unfortunately, these algorithms can be considered as being FPT only if Δ is part of the parameter. In Section 5.1, we present two linear fpt-reductions which prove that the problem is W[2]-hard when parameterised only by the size of the solution and W[1]-hard when parameterised only by the treewidth. These reductions also show that we can not even have an algorithm that computes $\mathrm{I}(G)$ in time $f(k)n^{o(k)}$ or $\mathcal{O}^*(f(tw)n^{o(tw)})$, unless the ETH fails. The $\mathcal{O}^*$ notation is used to suppress polynomial factors in regards to n and tw.

2 Preliminaries

For notions and definitions on graph theory not explained here, we refer the reader to [18].

Let $G = (V, E)$ be a graph and $G' = (V', E')$ be a subgraph of G (*i.e.*, created by deleting vertices and/or edges of G). Recall first that the subgraph G' is *induced* if it can be created only by deleting vertices of G. That is, for each edge $uv \in E$, if $u, v \in V'$, then $uv \in E'$. For any vertex $v \in V$, let $N_G(v) = \{u \in V : uv \in E\}$ denote the *neighbourhood* of v in G, and let $d_G(v) = |N_G(v)|$ denote the *degree* of v in G. We also define $N_G[v] = N_G(v) \cup \{v\}$. Finally, for any $X \subseteq V$, we define $N_G[X] = \bigcup_{v \in X} N_G[v]$. Note that, whenever the graph G is clear from the context, we will omit the subscript and simply write $N(v)$, $d(v)$, $N[v]$ and $N[X]$.

One way to show that a problem can not be approximated within a certain ratio, is through a *gap reduction*. The goal of such a reduction is to show that it is $\mathcal{NP}$-hard to differentiate between instances that have a solution of size $\leq \alpha$ and those for which any solution has size $> \beta$. If such is the case, then we know that we cannot approximate the optimal solution within a ratio of $\frac{\beta}{\alpha}$, as otherwise we would get that $\mathcal{P}$=$\mathcal{NP}$.

Finally, recall that a *fixed parameter-tractable* (FPT for short) algorithm, is an algorithm with running time $f(k)n^{\mathcal{O}(1)}$, where f is a computable function and k is the considered parameter. We also make use of what is known as a *linear fpt-reduction*, a type of polynomial reduction such that the size of the parameter of the new problem is linear in regards to the size of the parameter of the original problem. Observe that if we have a linear fpt-reduction from a problem Q with parameter k to a problem Q' with parameter k' and the assumption that Q can not be solved in time $f(k)n_1^{o(k)}$ (where n_1 is the size of the input of Q), then we can conclude that there is no $f(k')n_2^{o(k')}$ time algorithm for Q (where n_2 is the size of the input of Q).

Let $G = (V, E)$ be a graph. We say that G is *locally irregular* if for every edge $uv \in E$, we have $d(u) \neq d(v)$. Now, let $S \subseteq V$ be such that $G[V \setminus S]$ is a locally irregular graph; any set S that has this property is said to be an *irregulator of* G. For short, we will say that S is an $ir(G)$. Moreover, let $\mathrm{I}(G)$ be the minimum order that any $ir(G)$ can have. We will say that S is a *minimum* irregulator of G, for short S is an $ir^*(G)$, if S is an $ir(G)$ and $|S| = \mathrm{I}(G)$.

We also define the following notion, which generalises $ir(G)$. Let $G = (V, E)$ be a graph, $S, X \subseteq V$ and let $G' = G[V \setminus S]$. Now, let $S \subseteq V$ be such that, for each two neighbouring vertices u, v in $X \setminus S$, we have that $d_{G'}(u) \neq d_{G'}(v)$; any set S that has this property is said to be an *irregulator of* X *in* G, for short $ir(G, X)$. We define the notions of $ir^*(G, X)$ and $\mathrm{I}(G, X)$ analogously to the previous definitions.

We will now provide some lemmas and an observation that will be useful throughout this paper. As the proofs of the following lemmas mainly follow from the definitions, we chose to only include them in the full version of this paper. In the three lemmas below, we investigate the relationship between $\mathrm{I}(G)$ and $\mathrm{I}(G, X)$.

▶ **Lemma 1.** *Let $G = (V, E)$ be a graph and let $X \subseteq V$. Then $\mathrm{I}(G, X) \leq \mathrm{I}(G)$.*

▶ **Lemma 2.** *Let $G = (V, E)$ be a graph and $S, X \subseteq V$ such that S is an $ir^*(G, X)$. Then, $S \subseteq N[X]$ and $\mathrm{I}(G, X) = \mathrm{I}(G[N[X]], X)$.*

▶ **Lemma 3.** *Let $G = (V, E)$ be a graph, and $X_1, \dots, X_n \subseteq V$ such that $N[X_i] \cap N[X_j] = \emptyset$ for every $1 \leq i < j \leq n$. Then $\sum_{i=1}^{n} \mathrm{I}(G, X_i) \leq \mathrm{I}(G)$.*

▶ **Lemma 4.** *Let $G = (V, E)$ be a graph, X be a subset of V and S be an $ir(G)$. The set $S \cap N[X]$ is an $ir(G, X)$ and an $ir(G[N[X]], X)$.*

The following, almost trivial, observation, will be useful throughout the rest of the paper.

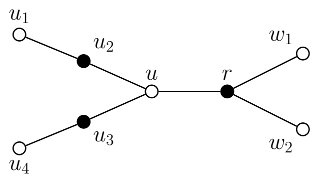


Figure 1 The gadget used in the proof of Theorem 7. The white and black vertices are used to denote vertices belonging to different bipartitions.

▶ **Observation 5.** *Let $G = (V, E)$ be a graph and S be an $ir(G)$. Then, for each edge $uv \in E$, if $d(u) = d(v)$, then S contains at least one vertex in $N[\{u, v\}]$. Additionally, for a set $X \subseteq V$, let S^* be an $ir(G[N[X]], X)$. Then for each edge $uv \in E(G[X])$, if $d(u) = d(v)$, then S^* contains at least one vertex in $N[\{u, v\}]$.*

3 (Classic) complexity

In this section, we deal with the complexity of the problem we introduced. In the following theorem, we sum up all the families of graphs for which we prove that $\mathrm{I}(G)$ is computed in polynomial time.

▶ **Theorem 6.** *Let G be a graph. If G is a path, cycle, tree, complete bipartite or a complete graph, then the problem of computing $\mathrm{I}(G)$ is in $\mathcal{P}$.*

The result for the case of paths and cycles is proven through induction on the order of the graph. Then, complete and complete bipartite graphs have a rather trivial structure in regards to the problem studied here. Finally, the polynomial algorithm for trees follows directly from upcoming Theorem 14.

3.1 $\mathcal{NP}$-Hard Cases

We now show that finding a minimum irregulator of a graph is $\mathcal{NP}$-hard. Interestingly, this remains true even for quite restricted families of graphs, such as cubic (i. e., 3-regular) bipartite, and subcubic planar bipartite graphs, i. e., planar bipartite graphs of maximum degree at most 3.

▶ **Theorem 7.** *Let G be graph and $k \in \mathbb{N}$. Deciding if $\mathrm{I}(G) \leq k$ is $\mathcal{NP}$-complete, even when G is a planar bipartite graph with maximum degree $\Delta \leq 3$.*

Proof. Since the problem is clearly in $\mathcal{NP}$, we will focus on proving it is also $\mathcal{NP}$-hard. The reduction is from the Vertex Cover problem, which remains $\mathcal{NP}$-complete when restricted to planar cubic graphs [27]. In that problem, a planar cubic graph G and an integer $k \geq 1$ are given as an input. The question is, whether there exists a vertex cover of G of order at most k. That is, whether there exists a set $VC \subseteq V(G)$ such that for every edge $uv \in E(G)$, at least one of u and v belongs in VC and $|VC| \leq k$.

Let G' be a planar cubic graph and $k \geq 1$ given as input for Vertex Cover. Let $|E(G')| = m$. We will construct a planar bipartite graph G as follows; we start with the graph G', and modify it by using multiple copies of the gadget, illustrated in Figure 1. Note that we will be following the naming convention illustrated in Figure 1 whenever we talk about the vertices of our gadgets. When we say that we *attach* a copy H of the gadget to the vertices v and v' of G', we mean that we add H to G', and we identify the vertices w_1 and w_2 to the vertices v and v' respectively. Now, for each edge $vv' \in E(G')$, attach one

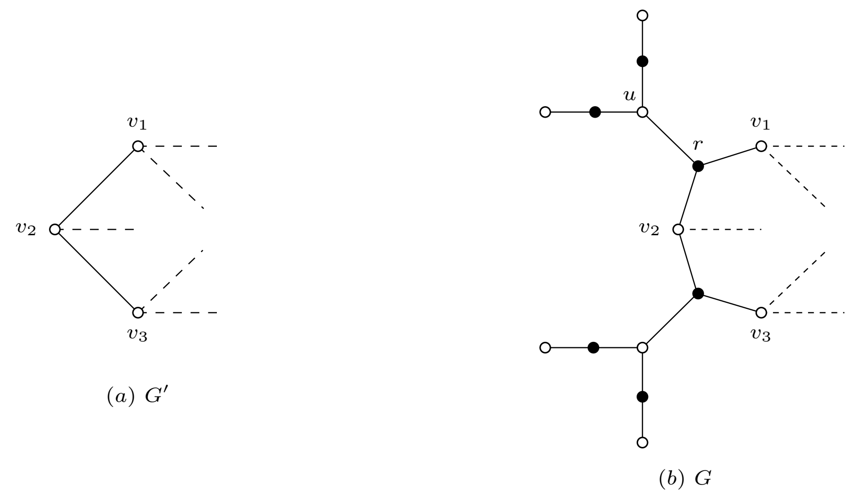


Figure 2 The construction in the proof of Theorem 7. The graph G' is the initial planar cubic graph, and G is the graph built during our reduction. In G, the white and black vertices are used to denote vertices belonging to different bipartitions.

copy H of the gadget to the vertices v and v', and then delete the edge vv' (see Figure 2). Clearly this construction is achieved in linear time (we have added m copies of the gadget). Note also that the resulting graph G has $\Delta(G) = 3$ and that the planarity of G' is preserved since G is constructed by essentially subdividing the edges of G' and adding a tree pending from each new vertex. Also, G is bipartite. Indeed, observe that after removing the edges of $E(G')$, the vertices of $V(G')$ form an independent set of G. Furthermore, the gadget is bipartite, and the vertices w_1, w_2 (that have been identified with vertices of $V(G')$) belong to the same bipartition (in the gadget). Finally, for any $1 \leq i \leq m$, let H_i be the i^{th} copy of the gadget attached to vertices of G'. We will also be using the vertices r^i and u^i to denote the copies of the vertices r and u (respectively) that also belong to H_i.

We are now ready to show that the minimum vertex cover of G' has size k' if and only if $\mathrm{I}(G) = k'$.

Let VC be a minimum vertex cover of G' and $|VC| = k'$. We will show that the set $S = VC$ is an $ir(G)$. Let $G^* = G[V(G) \setminus S]$. First, note that S contains only vertices of G'. Thus, for each i, the vertices of H_i except from r^i, which also remain in G^*, have the same degree in G' and in G^*. Also note that each vertex of G' is adjacent only to copies of r. It follows that it suffices to only consider the vertices r^i to show that VC is an $ir(G)$. Now, for any $1 \leq i \leq m$, consider the vertex r^i. Since VC is a vertex cover of G', for each edge $vv' \in E(G')$, VC contains at least one of v and v'. It follows that $d_{G^*}(r^i) \leq 2$. Note also that $N_{G^*}(r^i)$ contains the vertex $u^i \in V(H_i)$ and possibly one vertex $v \in V(G')$.

Also, since we only delete vertices in $V(H_i) \cap V(G')$, we have that $d_{G^*}(u^i) = 3 > d_{G^*}(r^i)$. In the case where $N_{G^*}(r^i)$ also contains a vertex $v \in V(G')$, the vertex v is adjacent only to vertices which do not belong in $V(G')$. Thus, $d_{G^*}(v) = d_G(v) = 3 > d_{G^*}(r^i)$. It follows that r^i has a different degree from all of its neighbours and that VC is an $ir(G)$.

Now, we prove that if $\mathrm{I}(G) = k'$ then there exists a vertex cover of size at most k'. Assume that $\mathrm{I}(G) = k'$ and let S be an $ir^*(G)$. Observe that since S is an $ir^*(G)$, S contains at least one vertex of H_i (for each $1 \leq i \leq m$). Let $X_i = V(H_i) \cap V(G')$. To construct a vertex cover VC of G' with $|VC| \leq k'$, we work as follows. For each $1 \leq i \leq m$:

1. for each vertex $v \in X_i$, if $v \in S$ then put v in VC. Then,
2. if $S \cap X_i = \emptyset$, put any one of the two vertices of X_i in VC.

Observe now that any vertex that is added to VC during step 1. of the above procedure, also belongs to S and any vertex that is added during step 2. of the above procedure corresponds to at least one vertex in S. It follows that $|VC| \leq k'$. Also note that VC contains at least one vertex of X_i, for each i, and that for each $uv \in E(G')$, there exists an i such that $V(X_i) = \{u, v\}$. Thus VC is indeed a vertex cover of G'.

Therefore G' has a minimum vertex cover of size k' if and only if $\mathrm{I}(G) = k'$. To complete the proof note that deciding if $\mathrm{I}(G) = k' < k$ for a given k, answers the question whether G' has a vertex cover of size less than k or not. ◀

In the following theorem we show that calculating $\mathrm{I}(G)$ is $\mathcal{NP}$-hard even if G is a cubic bipartite graph.

▶ **Theorem 8.** *Let G be graph and $k \in \mathbb{N}$. Deciding if $\mathrm{I}(G) \leq k$ is $\mathcal{NP}$-complete even in cubic bipartite graphs.*

This theorem is shown through a reduction from the 2-Balanced 3-SAT, which was proven to be $\mathcal{NP}$-complete in [9].

4 (In)approximability

In the previous section we showed that computing $\mathrm{I}(G)$ is $\mathcal{NP}$-hard, even for graphs G belonging to quite restricted families of graphs. So the natural question to pose next, which we investigate in this section, is whether we can approximate $\mathrm{I}(G)$. Unfortunately, most of the results we present below are once again negative.

We start with a corollary that follows from the proof of Theorem 7 and the inapproximability of Vertex Cover in cubic graphs [17]:

▶ **Corollary 9.** *Given a graph G, it is $\mathcal{NP}$-hard to approximate $\mathrm{I}(G)$ to within a ratio of $\frac{100}{99}$, even if G is bipartite and $\Delta(G) = 3$.*

Now, we are going to show that there can be no algorithm that approximates $\mathrm{I}(G)$ to within any decent ratio in polynomial time, unless $\mathcal{P}$=$\mathcal{NP}$, even if G is a bipartite graph (with no restriction on its maximum degree).

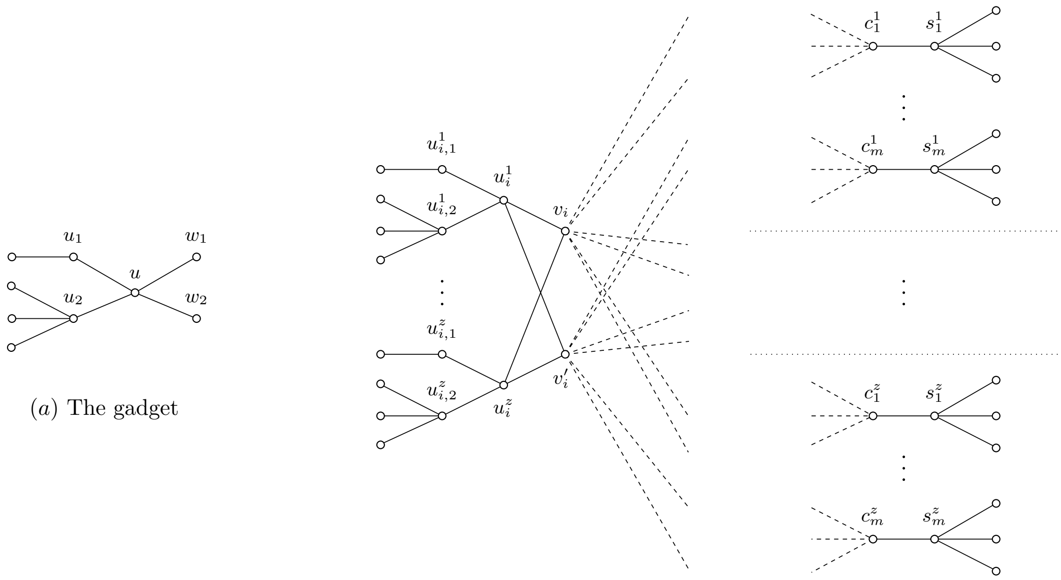
▶ **Theorem 10.** *Let G be a bipartite graph of order N and $k \in \mathbb{N}$ be a constant such that $k \geq 1$. It is $\mathcal{NP}$-hard to approximate $\mathrm{I}(G)$ to within $\mathcal{O}(N^{1-\frac{1}{k}})$.*

Proof. The proof is by a *gap producing reduction* from 2-Balanced 3-SAT, which was proven to be $\mathcal{NP}$-complete in [9]. In that problem, a 3CNF formula F is given as an input, comprised by a set C of clauses over a set of Boolean variables X. In particular, we have that each clause contains exactly 3 literals, and each variable $x \in X$ appears in F exactly twice as a positive and twice as a negative literal. The question is, whether there exists a truth assignment to the variables of X satisfying F.

Let F be a 3CNF formula with m clauses $C_1, \dots, C_m$ and n variables $x_1, \dots, x_n$ that is given as input to the 2-Balanced 3-SAT problem. Let $2k = k' + 1$. Based on the instance F, we are going to construct a bipartite graph $G = (V, E)$ where $|V| = \mathcal{O}(n^{k'+1})$ and

- $\mathrm{I}(G) \leq n$ if F is satisfiable
- $\mathrm{I}(G) > n^{k'}$ otherwise.

To construct $G = (V, E)$, we start with the following graph: for each literal x_i ($\neg x_i$ resp.) in F, add a *literal vertex* v_i (v'_i resp.) in V, and for each clause C_j of F, add a *clause vertex* c_j in V. Next, for each $1 \leq j \leq m$, add the edge $v_i c_j$ ($v'_i c_j$ resp.) if the literal x_i ($\neg x_i$ resp.)



(a) The gadget

(b) The constructed graph G for $n^{k'} = z$

Figure 3 The construction in the proof of Theorem 10. In subfigure (b), we illustrate how each pair of literal vertices is connected to the rest of the graph. Whenever there is an upper index $1 \le l \le n^{k'}$ on a vertex, it is used to denote the l^{th} copy of that vertex. The dashed lines are used to represent the edges between the literal and the clause vertices.

appears in C_j according to F. Observe that the resulting graph is bipartite, for each clause vertex c we have $d(c) = 3$ and for each literal vertex v we have $d(v) = 2$ (since in F, each variable appears twice as a positive and twice as a negative literal). To finish the construction of G, we will make use of the gadget shown in Figure 3(a), as well as some copies of S_5, the star on 5 vertices. When we say that we *attach* a copy H of the gadget to the vertices v_i and v_i' (for some $1 \le i \le n$), we mean that we add H to G, and we identify the vertices w_1 and w_2 to the vertices v_i and v_i' respectively. Now:

- for each $1 \le i \le n$, we attach $n^{k'}$ copies of the gadget to the vertices v_i and v_i' of G. For convenience, we will give unique names to the vertices corresponding to each gadget added that way. So, the vertex u_i^l (for $1 \le l \le n^{k'}$ and $1 \le i \le n$) is used to represent the vertex u of the l^{th} copy of the gadget attached to v_i and v_i', and $u_{i,1}^l$ ($u_{i,2}^l$ resp.) is used to denote the vertex u_1 (u_2 resp.) of that same gadget. Then,
- for each $1 \le j \le m$, we add $n^{k'} - 1$ copies of the clause vertex c_j to G, each one of these copies being adjacent to the same literal vertices as c_j. For $1 \le l \le n^{k'}$, the vertex c_j^l is the l^{th} copy of c_j. Finally,
- for each $1 \le j \le m$ and $1 \le l \le n^{k'}$, we add a copy of the star on 5 vertices S_5 to G and identify any degree-1 vertex of S_5 to c_j^l. Let s_j^l be the neighbour of c_j^l that also belongs to a copy of S_5.

Observe that the resulting graph G (illustrated in Figure 3(b)) remains bipartite and that this construction is achieved in polynomial time in regards to $n + m$.

From the construction of G, we know that for every $1 \le i \le n$, $d(v_i) = d(v_i') = \Theta(n^{k'})$. So, for sufficiently large n, the only pairs of adjacent vertices of G that have the same degrees are either the vertices u_i^l and $u_{i,2}^l$, or the vertices c_j^l and s_j^l (for every $1 \le i \le n$, $1 \le l \le n^{k'}$ and $1 \le j \le m$).

First, let F be a satisfiable formula and let t be a satisfying assignment of F. Also, let S be the set of literal vertices v_i (v'_i resp.) such that the corresponding literals x_i ($\neg x_i$ resp.) are assigned value *true* by t. Clearly $|S| = n$. We will also show that S is an $ir(G)$. Consider the graph $G' = G[V \setminus S]$. Now, for any $1 \leq i \leq n$, we have that either v_i or v'_i, say v_i, belongs to the vertices of G'. Now for every $1 \leq l \leq n^k$, we have that $d_{G'}(u^l_i) = 3$, while $d_{G'}(u^l_{i,1}) = 2$ and $d_{G'}(u^l_{i,2}) = 4$ (since none of the neighbours of $u^l_{i,1}$ and $u^l_{i,2}$ belongs to S). Also, for every $1 \leq j \leq m$ and $1 \leq l \leq n^{k'}$, since t is a satisfying assignment of F, $N(c^l_j)$ contains at least one vertex in S. It follows that $d_{G'}(c^l_j) = 3 < 4 = d_{G'}(s^l_j)$. Finally, since S does not contain any neighbours of v_i, we have that $d_{G'}(v_i) = d_G(v_i) = \mathcal{O}(n^{k'})$. It follows that S is an $ir(G)$ and thus that $\mathrm{I}(G) \leq n$.

Now let F be a non-satisfiable formula and assume that there exists an S that is an $ir(G)$ with $|S| \leq n^{k'}$. As usual, let $G' = G[V \setminus S]$. Then:

1. For every $1 \leq j \leq m$, there exists a literal vertex v such that $v \in N(c^l_j)$ for every $1 \leq l \leq n^{k'}$. Assume that this is not true for a specific j. Then, since $d_G(c^l_j) = d_G(s^l_j) = 4$, for every $1 \leq l \leq n^{k'}$, we have that S contains at least one vertex in $N[\{c^l_j, s^l_j\}]$, which does not belong to the literal vertices. That is, S contains at least one (non-literal) vertex for each one of the $n^{k'}$ copies of c_j. Observe also that even if this is the case, S would also have to contain at least one more vertex to, for example, stop $u^1_{i,2}$ and u^1_i, from having the same degree in G'. It follows that $|S| > n^{k'}$, which is a contradiction.
2. For every $1 \leq i \leq n$, S does not contain both v_i and v'_i. Assume this is not true for a specific i. Then, for every $1 \leq l \leq n^{k'}$, we have that $d_{G'}(u^l_i) = d_{G'}(u^l_{i,1}) = 2$, unless S also contains an additional vertex of the gadgets attached to v_i and v'_i, for each one of the $n^{k'}$ such gadgets. It follows that $|S| \geq n^{k'}$. Since we have also assumed that for a specific i, both v_i and v'_i belong to S, we have that $|S| > n^{k'}$, a contradiction.
3. For every $1 \leq i \leq n$, S contains at least one of v_i and v'_i. Assume this is not true for a specific i. Then, for every $1 \leq l \leq n^{k'}$, we have that $d_{G'}(u^l_i) = d_{G'}(u^l_{i,2}) = 4$, unless S also contains an additional vertex of the gadgets attached to v_i and v'_i, for each one of the $n^{k'}$ such gadgets. Even if this is the case, S would also have to contain at least one more vertex to, for example, stop c^1_1 and S^1_1 from having the same degree in G'. It follows that $|S| > n^{k'}$, which is a contradiction.

So from items 2. and 3. above, it follows that for each $1 \leq i \leq n$, S contains exactly one of v_i and v'_i. Now consider the following truth assignment: we assign the value *true* to every variable x_i if the corresponding literal vertex v_i belongs in S, and value *false* to every other variable. Now, from item 1. above, it follows that each clause C_j contains either a positive literal x_i which has been set to *true*, or a negative literal $\neg x_i$ which has been set to *false*. Thus F is satisfied, which is a contradiction.

Up to this point, we have shown that there exists a graph $G = (V, E)$ with $|V(G)| = N = O(n^{k'+1})$ where

- $\mathrm{I}(G) \leq n$ if F is satisfiable
- $\mathrm{I}(G) > n^{k'}$ otherwise.

Therefore, we have that $\mathrm{I}(G)$ is not $\mathcal{O}(n^{k'-1})$ approximable in polynomial time unless $\mathcal{P}$=$\mathcal{NP}$.

Now, since $N = |V(G)| = \Theta(n^{k'+1})$ and $2k = k' + 1$ we have $\mathcal{O}(n^{k'-1}) = \mathcal{O}(N^{\frac{k'-1}{k'+1}}) = \mathcal{O}(N^{1-\frac{2}{k'+1}}) = \mathcal{O}(N^{1-\frac{1}{k}})$. This ends the proof of this theorem. ◀

Now, we consider the case where G is regular bipartite graph. Below we present an upper bound to the size of $\mathrm{I}(G)$. This upper bound is then used to obtain a (simple) Δ-approximation of an optimal solution.

▶ **Theorem 11.** *For any d-regular bipartite graph $G = (L, R, E)$ of order n we have that* $\mathrm{I}(G) \geq n/2d$.

Now recall that in any bipartite graph G, any bipartition of G is a vertex cover of G. Also observe that any vertex cover of a graph G, is also an irregulator of G. Indeed, deleting the vertices of any vertex cover of G, leaves us with an independent set, which is locally irregular. The next corollary follows from these observations and Theorem 11:

▶ **Corollary 12.** *For any d-regular bipartite graph $G = (L, R, E)$, any of the sets L and R is a d-approximation of $ir^*(G)$.*

5 Parameterised complexity

As the problem of computing a minimal irregulator of a given graph G seems to be rather hard to solve, and even to approximate, we focused our efforts towards finding parameterised algorithms that can solve it. First we present an FPT algorithm that calculates $\mathrm{I}(G)$ when parameterised by the size of the solution and Δ, the maximum degree of the graph.

▶ **Theorem 13.** *For a given graph $G = (V, E)$ with $|V| = n$ and maximum degree Δ, and for $k \in \mathbb{N}$, there exists an algorithm that decides if $\mathrm{I}(G) \leq k$ in time $(2\Delta)^k n^{\mathcal{O}(1)}$.*

The main tool we use to show Theorem 13 is Observation 5. Let $G = (V, E)$ be a graph and $k \in \mathbb{N}$. A high level description of our recursive algorithm is as follows: first find an edge $uv \in E$ such that $d(u) = d(v)$. Now, assume that we are making a correct guess of a vertex $w \in N[\{u, v\}] \cap S$ where S is a minimum irregulator. Then, $G_w = G[V \setminus w]$ must have a minimum irregulator of size $|S| - 1$. Note that if we repeat the above process and we make correct guesses, we are going to stop after deleting $|S|$ vertices or when we have deleted k vertices (meaning that $\mathrm{I}(G) > k$). Then, by considering all the 2Δ choices for w, we have a running time of $(2\Delta)^k$.

We now turn our attention towards graphs that are "close to being trees", that is graphs of bounded treewidth. In particular, we provide an FPT algorithm that finds a minimum irregulator of G, when parameterised by the treewidth of the input graph and by Δ.

▶ **Theorem 14.** *For a given a graph $G = (V, E)$ and a nice tree decomposition of G, there exists an algorithm that returns $\mathrm{I}(G)$ in time $\Delta^{2tw} n^{\mathcal{O}(1)}$, where tw is the treewidth of the given decomposition and Δ is the maximum degree of G.*

The idea of the proof of Theorem 14, is based on the classic dynamic programming technique on the given nice tree decomposition of G. Let us denote by B_c the bag of vertices of a node c of a nice tree decomposition of G. In essence, for each node c of the tree decomposition, we store the necessary information that allows us to find all the sets that are $ir(G, B_c^{\downarrow} \setminus B_c)$, where $B_c^{\downarrow}$ denotes the vertices appearing in a sub-tree rooted at c. Then for the root r of the tree decomposition, we can check which of the stored sets that are $ir(G, B_r^{\downarrow} \setminus B_r)$, are also $ir(G)$; the minimum such set is an $ir^*(G)$.

The running time of our algorithm follows from the size of the tables we keep for these sets. In particular, for each set stored for a node c, for each vertex v of B_c, we keep the degree that we want v to have in the final, locally irregular graph (i. e. the graph G after the removal of $ir(G)$) and the degree that v has in $G[B_c^{\downarrow} \setminus S]$. This gives us Δ^2 choices for each vertex of B_c.

It is worth noting that the algorithms of Theorem 13 and 14 can be used in order to also return an $ir^*(G)$.

5.1 W-Hardness

Observe that both of the algorithms presented above, have to consider Δ as part of the parameter if they are to be considered as FPT. The natural question to ask at this point is whether we can have an FPT algorithm, when parameterised only by the size of the solution, or the treewidth of the input graph. In this section, we give a strong indication towards the negative answer for both cases, proving that, in some sense, the algorithms provided in Section 5 are optimal.

▶ **Theorem 15.** *Let G be a graph and $k \in \mathbb{N}$. Deciding if $\mathrm{I}(G) \leq k$ is $\mathrm{W}[2]$-hard, when parameterised by k.*

The proof of Theorem 15 is done through a linear-fpt reduction from the Dominating Set problem, when parameterised by the size of the solution.

▶ **Theorem 16.** *Let G be a graph with treewidth tw, and $k \in \mathbb{N}$. Deciding if $\mathrm{I}(G) = k$ is $W[1]$-hard when parameterised by tw.*

Proof. We will present a reduction from the List Colouring problem: the input consists of a graph $H = (V, E)$ and a list function $L : V \to \mathcal{P}(\{1, \dots, k\})$ that specifies the available colours for each vertex $u \in V$. The goal is to find a proper colouring $c : V \to \{1, \dots, k\}$ such that $c(u) \in L(u)$ for all $u \in V$. When such a colouring exists, we say that (H, L) is a *yes-instance* of List Colouring. This problem is known to be $W[1]$-hard when parameterised by the treewidth of H [19].

Now, starting from an instance (H, L) of List Colouring, we will construct a graph $G = (V', E')$ (see Figure 4 (a)) such that:

- $|V'| = \mathcal{O}(|V|^6)$,
- $tw(G) = tw(H)$ and
- $\mathrm{I}(G) = nk$ if and only if (H, L) is a yes-instance of List Colouring.

Before we start with the construction of G, let us give the following observation.

▶ **Observation 17.** *Let (H, L) be an instance of List Colouring where $H = (V, E)$ and there exists a vertex $u \in V$ such that $|L(u)| > d(u)$. Then the instance $(H[V \setminus \{u\}], L')$, where $L'(v) = L(v)$ for all $v \in V \setminus \{u\}$, is a yes-instance of List Colouring if and only if (H, L) is a yes-instance of List Colouring.*

Indeed, observe that for any vertex $u \in V$, by any proper colouring c of H, $c(u)$ only has to avoid $d(u)$ colours. Since $|L(u)| > d(u)$, we will always have a spare colour to use on u that belongs in $L(u)$. From the previous observation, we can assume that in our instance, for all $u \in V$, we have $|L(u)| \leq d(u)$. Furthermore, we can deduce that $k \leq n(n-1)$ as the degree of any vertex is at most $n-1$. Finally, let us denote by $\overline{L}(u)$ the set $\{0, 1, \dots, k\} \setminus L(u)$. It is important to note here that for every $u \in V$, the list $L(u)$ contains at least one element belonging in $\{1, \dots, k\}$. It follows that $\overline{L}(u)$ also contains at least one element, the colour 0. To sum up, we have that $1 \leq |\overline{L}(u)| \leq k$.

Now, we present the three gadgets we are going to use in the construction of G. First, we have the “forbidden colour gadget” H_i, which is a star with i leaves (see Figure 4(c)). When we say that we attach a copy of H_i on a vertex v of a graph G, we mean that we add H_i to G and we identify the vertices v and w_2 (where here and in what follows, we are using the naming illustrated in Figure 4 when talking about the vertices w_1, w_2, w_3, v_1 and v_2). The second, will be the “degree gadget”, which is presented in Figure 4(b). Finally, we have the “horn gadget”, which is a path on three vertices (see Figure 4(d)). We define the

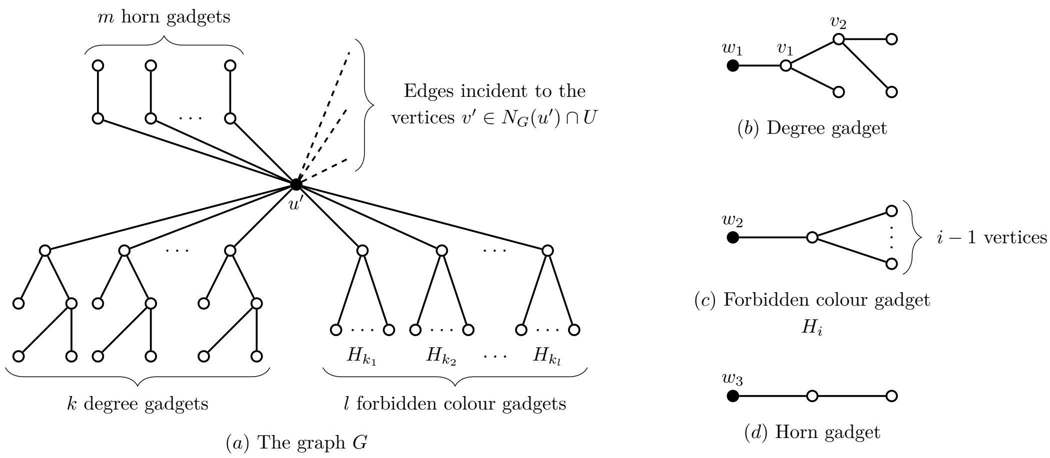


Figure 4 In (a) we illustrate the construction of G, as it is described in the proof of Theorem 16. The black vertex represents every vertex that belongs in U. For the specific vertex u' shown in the figure, we have that $\overline{L}(u) = \{c_1, \dots, c_l\}$ and $k_i = n^3 - c_i$ for all $i = 1, \dots, l$. We also have that $m = 2n^3 - d_G(u) - k - l$.

operation of attaching these two gadgets on a vertex v of a graph G similarly to how we defined this operation for the forbidden colour gadget (each time using the appropriate w_1 or w_3, according to if it is a degree or a horn gadget respectively).

In order to construct G, we start from a copy of H. Let us use $G|_H$ to denote the copy of H that lies inside of G and, for each vertex $u \in V$, let u' be its copy in V'. We will call the set of these vertices U. That is, $U = \{v \in V(G|_H)\}$. Then, we are going to attach several copies of each gadget to u', for each vertex $u' \in U$. We start by attaching k copies of the degree gadget to each vertex $u' \in U$. Then, for each $u \in V$ and each $i \in \overline{L}(u)$, we attach one copy of the forbidden colour gadget H_{2n^3-i} to the vertex u'. Finally, for each $u' \in U$, we attach to u' as many copies of the horn gadget as are needed, in order to have $d_G(u') = 2n^3$.

Before we continue, observe that, for sufficiently large n, we have attached more than n^3 horn gadgets to each vertex of U. Indeed, before attaching the horn gadgets, each vertex $u' \in U$ has $d_G(u) \le n-1$ neighbours in U, k neighbours from the degree gadgets and at most $k < n^2$ neighbours from the forbidden colour gadgets (recall that $|\overline{L}(u)| \le k$). We will now show that $|V'| = \mathcal{O}(n^6)$. For that purpose, let us calculate the number of vertices in all the gadgets attached to a single vertex $u' \in U$. First, we have $5k < 5n^2$ vertices in the degree gadgets. Then, we have less than $4n^3$ vertices in the horn gadgets (as we have less that $2n^3$ such gadgets). Finally, we have at most $k < n^2$ forbidden colour gadgets, each one of which containing at most $2n^3$ vertices. So, for each vertex $u' \in U$, we have at most $2n^5 + 4n^3 + 5n^2$ vertices in the gadgets attached to u'. Therefore, we have $|V'| = \mathcal{O}(n^6)$.

Before we prove that $\mathrm{I}(G) \le nk$ if and only if (H, L) is a yes-instance of List Colouring, we need to argue about two things. First, about the treewidth of the graph G and second, about the minimum value of $\mathrm{I}(G)$. Since our construction only attaches trees to each vertex of $G|_H$ (and recall that a tree has a treewidth of 1 by definition), we know that $tw(G) = tw(G|_H) = tw(H)$. As for $\mathrm{I}(G)$, we will show that it has to be at least equal to nk. For that purpose we have the following two claims.

▷ Claim 18. Let S be an $ir(G)$ and $S \cap U \neq \emptyset$. Then $|S| > n^3$.

▷ Claim 19. Let S be an $ir(G)$ and $S \cap U = \emptyset$. Then $|S| \ge nk$. In particular, S includes at least one vertex from each copy of the degree gadget used in the construction of G.

By the previous two claims, we conclude that $\mathrm{I}(G) \geq nk$. We are ready to show that, if (H, L) is a yes-instance of List Colouring, then there exists a set $S \subseteq V'$ such that S is an $ir(G)$ and $|S| = nk$. Let c be a proper colouring of H such that $c(u) \in L(u)$ for all $u \in V$. We will construct an $ir(G)$ as follows. For each $u \in V$, we partition (arbitrarily) the k degree gadgets attached to the vertex u' to $c(u)$ "good" and $(k - c(u))$ "bad" degree gadgets. For each good degree gadget, we add the copy of the vertex v_1 of that gadget to S and for each bad degree gadget we add the copy of the vertex v_2 of that gadget to S. This process creates a set S of size nk, as it includes k distinguished vertices for each vertex $u' \in U$.

Now we need to show that S is an $ir(G)$. Let $G' = G[V' \setminus S]$; observe that each vertex $u' \in U$ has degree $d_{G'}(u') = 2n^3 - c(u)$. Therefore, u' does not have the same degree as any of its neighbours that do not belong in U. Indeed, for every $v \in N_{G'}(u') \setminus U$, we have that $d_{G'}(v) \in \{1, 2\}$ (if v belongs to a bad degree or a horn gadget) or $d_{G'}(v) \in \{2n^3 - i : i \in \overline{L}(u)\}$ (if v belongs to a forbidden colour gadget). Furthermore, since c is a proper colouring of H, for all $uv \in E$, we have that $c(u) \neq c(v)$. This gives us that for any edge $u'v' \in E'$ with $u', v' \in U$, we have that $d_{G'}(u') = 2n^3 - c(u) \neq 2n^3 - c(v) = d_{G'}(v')$.

So, we know that for every vertex $u' \in U$, there is no vertex $w \in N_{G'}(u')$ such that $d_{G'}(u') = d_{G'}(w)$. It remains to show that, in G', there exist no two vertices belonging to the same gadget, which have the same degrees. First of all, we have that S does not contain any vertex from any of the horn and forbidden colour gadgets, nor from U. Thus any adjacent vertices belonging to these gadgets have different degrees. Last, it remains to check the vertices of the degree gadgets. Observe that for any copy of the degree gadget, S contains either v_1 or v_2. In both cases, after the deletion of the vertices of S, any adjacent vertices belonging to any degree gadget have different degrees. Therefore, S is an $ir(G)$ of order nk and since $\mathrm{I}(G) \geq nk$ we have that $\mathrm{I}(G) = nk$.

Now, for the opposite direction, assume that there exists a set $S \subseteq V'$ such that S is an $ir^*(G)$ and $|S| = nk$. Let $G' = (V'', E'')$ be the graph $G[V' \setminus S]$. It follows from Claim 18 and Claim 19, that $S \cap U = \emptyset$ and that S contains exactly one vertex from each copy of the degree gadget in G and no other vertices. Consider now the colouring c of H defined as $c(u) = 2n^3 - d_{G'}(u')$. We will show that c is a proper colouring for H and that $c(u) \in L(u)$. First, we have that c is a proper colouring of H. Indeed, for any edge $uv \in E$, there exists an edge $u'v' \in E''$ (since $S \cap U = \emptyset$). Since G' is locally irregular we have that $d_{G'}(u') \neq d_{G'}(v')$, an thus $c(u) \neq c(v)$. It remains to show that $c(u) \in L(u)$ for all $u \in V$. First observe that, during the construction of G, we attached exactly k degree gadgets to each $u' \in U$. It follows that $d_{G'}(u') = 2n^3 - j$ and $c(u) = j$ for a $j \in \{0, 1, \ldots, k\}$. It is sufficient to show that $j \notin \overline{L}(u)$. Since S contains only vertices from the copies of the degree gadgets, we have that each $u' \in U$ has exactly one neighbour of degree $2n^3 - i$ for each $i \in \overline{L}(u)$ (this neighbour is a vertex of the H_i forbidden colour gadget that was attached to u'). Furthermore, for all $u' \in U$, since G' is locally irregular, we have that $d_{G'}(u') \neq 2n^3 - i$ for all $i \in \overline{L}(u)$. Equivalently, $d_{G'}(u') = 2n^3 - j$ for any $j \in L(u)$. Thus, $c(u) \in L(u)$ for all $u \in V$. ◀

Note that the reductions presented in the proofs of Theorem 15 and Theorem 16 are linear fpt-reductions. Additionally we know that

- there is no algorithm that answers if a graph G of order n has a Dominating Set of size at most k in time $f(k)n^{o(k)}$ unless the ETH fails [26] and
- there is no algorithm that answers if an instance (G, L) of the List Colouring is a yes-instance in time $\mathcal{O}^*(f(tw)n^{o(tw)})$ unless the ETH fails [19].

So, the following corollary holds.

▶ **Corollary 20.** *Let G be a graph of order n and assume the ETH. For $k \in \mathbb{N}$, there is no algorithm that decides if $\mathrm{I}(G) \leq k$ in time $f(k)n^{o(k)}$. Furthermore, assuming that G has treewidth tw, there is no algorithm that computes $\mathrm{I}(G)$ in time $\mathcal{O}^*(f(tw)n^{o(tw)})$.*

6 Conclusion

In this work we introduce the problem of identifying the largest locally irregular induced subgraph of a given graph. There are many interesting directions that could be followed for further research. An obvious one is to investigate whether the problem of calculating $\mathrm{I}(G)$ remains $\mathcal{NP}$-hard for other, restricted families of graphs. The first candidate for such a family would be the one of chordal graphs. On the other hand, there are some interesting families, for which the problem of computing an optimal irregulator could be decided in polynomial time, such as split graphs. Also, it could be feasible to conceive approximation algorithms for regular bipartite graphs, which have a better approximation ratio than the (simple) algorithm we present. The last aspect we find intriguing, is to study the parameterised complexity of calculating $\mathrm{I}(G)$ when considering other parameters, like the size of the minimum vertex cover of G, with the goal of identifying a parameter that suffices, by itself, in order to have an FPT algorithm. Finally, it is worth investigating whether calculating $\mathrm{I}(G)$ could be done in FPT time (parameterised by the size of the solution) in the case where G is a planar graph.

References

1 Agostinho Agra, Geir Dahl, Torkel Andreas Haufmann, and Sofia J. Pinheiro. The k-regular induced subgraph problem. *Discretete Applied Mathematics*, 222:14–30, 2017. doi:10.1016/j.dam.2017.01.029.

2 Yousef Alavi, Alfred Boals, Gary Chartrand, Ortrud Oellermann, and Paul Erdős. K-path irregular graphs. *Congressus Numerantium*, 65, January 1988.

3 Yousef Alavi, Gary Chartrand, Fan R. K. Chung, Paul Erdös, Ronald L. Graham, and Ortrud R. Oellermann. Highly irregular graphs. *Journal of Graph Theory*, 11(2):235–249, 1987. doi:10.1002/jgt.3190110214.

4 Akhbar Ali, Gary Chartrand, and Ping Zhang. *Irregularity in Graphs*. Springer briefs in mathematics. Springer, 2021. doi:10.1007/978-3-030-67993-4.

5 Yuichi Asahiro, Hiroshi Eto, Takehiro Ito, and Eiji Miyano. Complexity of finding maximum regular induced subgraphs with prescribed degree. *Theoretical Computer Science*, 550:21–35, 2014. doi:10.1016/j.tcs.2014.07.008.

6 Olivier Baudon, Julien Bensmail, Jakub Przybyło, and Mariusz Wozniak. On decomposing regular graphs into locally irregular subgraphs. *European Journal of Combinatorics*, 49:90–104, 2015. doi:10.1016/j.ejc.2015.02.031.

7 Rémy Belmonte and Ignasi Sau. On the complexity of finding large odd induced subgraphs and odd colorings. *Algorithmica*, 83(8):2351–2373, 2021. doi:10.1007/s00453-021-00830-x.

8 Julien Bensmail, Martin Merker, and Carsten Thomassen. Decomposing graphs into a constant number of locally irregular subgraphs. *European Journal of Combinatorics*, 60:124–134, 2017. doi:10.1016/j.ejc.2016.09.011.

9 Piotr Berman, Marek Karpinski, and Alex D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. *Electronic Colloquium on Computational Complexity*, 2003. URL: https://eccc.weizmann.ac.il/eccc-reports/2003/TR03-049/index.html.

10 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

11 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.

12 Gary Chartrand, Paul Erdös, and Ortrud Oellermann. How to define an irregular graph. *The College Mathematics Journal*, 19, January 1988. doi:10.2307/2686701.

13 Gary Chartrand, Michael Jacobon, Jenö Lehel, Ortrud Oellermann, Sergio Ruiz, and Farrokh Saba. Irregular networks. *Congressus Numerantium*, 64, January 1986.

14 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized np-hard problems. *Information and Computation*, 201(2):216–231, 2005. doi:10.1109/CCC.2004.1313826.

15 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. On the computational hardness based on linear FPT-reductions. *Journal of Combinatorial Optimization*, 11(2):231–247, 2006. doi:10.1007/s10878-006-7137-6.

16 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

17 Miroslav Chlebík and Janka Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theor. Comput. Sci.*, 354(3):320–338, 2006. doi:10.1016/j.tcs.2005.11.029.

18 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. doi:10.1007/978-3-662-53622-3.

19 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

20 Alan M. Frieze, Ronald J. Gould, Michal Karonski, and Florian Pfender. On graph irregularity strength. *Journal of Graph Theory*, 41(2):120–137, 2002. doi:10.1002/jgt.10056.

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

22 Michał Karoński, Tomasz Łuczak, and Andrew Thomason. Edge weights and vertex colors. *Journal of Combinatorial Theory*, 91:151–157, May 2004. doi:10.1016/j.jctb.2003.12.001.

23 Subhash Khot and Venkatesh Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, 289(2):997–1008, 2002. doi:10.1016/S0304-3975(01)00414-5.

24 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

25 Carla Negri Lintzmayer, Guilherme Oliveira Mota, and Maycon Sambinelli. Decomposing split graphs into locally irregular graphs. *Discrete Applied Mathematics*, 292:33–44, 2021. doi:10.1016/j.entcs.2019.08.053.

26 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the European Association for Theoretical Computer Science*, 105:41–72, 2011. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/92.

27 Bojan Mohar. Face covers and the genus problem for apex graphs. *J. Comb. Theory, Ser. B*, 82(1):102–117, 2001. doi:10.1006/jctb.2000.2026.

28 Hannes Moser and Dimitrios M. Thilikos. Parameterized complexity of finding regular induced subgraphs. *Journal of Discrete Algorithms*, 7(2):181–190, 2009. doi:10.1016/j.jda.2008.09.005.

29 Jakub Przybyło. Irregularity strength of regular graphs. *Electronic Journal of Combinatorics*, 15, June 2008. doi:10.37236/806.

30 Jakub Przybyło. On decomposing graphs of large minimum degree into locally irregular subgraphs. *Electronic Journal of Combinatorics*, 23(2):2–31, 2016. doi:10.37236/5173.

A Omitted proofs

A.1 Proof of Theorem 13

Let us first present the following lemma:

▶ **Lemma 21.** *Let $G = (V, E)$ be a graph such that, G is not locally irregular, and S be an $ir^*(G)$. Furthermore let $G_v = (V', E')$ be the graph $G[V \setminus \{v\}]$ for a vertex $v \in S$. Then* $\mathrm{I}(G_v) = \mathrm{I}(G) - 1$.

Proof. First observe that $S' = S \setminus \{v\}$ must be an $ir(G_v)$ as $G_v[V' \setminus S'] = G[V \setminus S]$. It follows that $\mathrm{I}(G_v) \leq \mathrm{I}(G) - 1$. Assume that $\mathrm{I}(G_v) < \mathrm{I}(G) - 1$. Then these exists an S'' such that $|S''| < \mathrm{I}(G) - 1$ and S'' is an $ir(G_v)$. Since $G_v[V' \setminus S''] = G[V \setminus (S'' \cup \{v\})]$, we have that $S'' \cup \{v\}$ is an $ir(G)$ and $|S'' \cup \{v\}| = |S''| + 1 < \mathrm{I}(G)$. This is a contradiction. ◀

Now, we are ready to present the proof of the theorem.

Proof of Theorem 13. In order to decide if $\mathrm{I}(G) \leq k$ we are going to use a recursive algorithm. The algorithm has input (G, k), where $G = (V, E)$ is a graph and $k \geq 0$ is an integer. The basic idea of this algorithm, is to take advantage of Observation 5. We present the exact procedure in Algorithm 1.

Algorithm 1 [IsIrregular(G, k) decision function].

Input: A graph $G = (V, E)$ and an integer $k \geq 0$.
Output: Is $\mathrm{I}(G) \leq k$ or not?
1: **if** G is irregular **then**
2:     **return** *yes*
3: **else if** $k = 0$ **then**
4:     **return** *no*
5: **else** ▷ $k > 0$ and G is not irregular
6:     $ans \leftarrow no$
7:     find an edge $vu \in E$ such that $d_G(v) = d_G(u)$
8:     **for all** $w \in N_G[\{u, v\}]$ **do**
9:         set $G_w = G[V \setminus \{w\}]$
10:         **if** IsIrregular$(G_w, k-1)$ returns *yes* **then**
11:             $ans \leftarrow yes$
12:     **return** ans

Now, let us argue about the correctness and the efficiency of this algorithm. We claim that for any graph $G = (V, E)$ and any integer $k \geq 0$, Algorithm 1 returns yes if $\mathrm{I}(G) \leq k$ and no otherwise. Furthermore, the number of steps that the algorithm requires, is $f(k, n) = (2\Delta)^k n^{\mathcal{O}(1)}$, where $n = |V|$. We will prove this by induction on k.

Base of the induction $(k = 0)$: Here, we only need to check if G is locally irregular. Algorithm 1 does this in line 1 and returns yes if it is (line 2) and no otherwise (line 4). Furthermore, we can check if G is locally irregular in polynomial time. So, the claim is true for the base.

Induction hypothesis $(k = k_0 \geq 0)$: We assume that we have a $k_0 \geq 0$ such that Algorithm 1 can decide if any graph G with n vertices and maximum degree Δ has $\mathrm{I}(G) \leq k_0$ in $f(k_0, n) = (k_0 + 1)(2\Delta)^{k_0} n^{\mathcal{O}(1)}$ steps.

Induction step ($k = k_0 + 1$): Let $G = (V, E)$ be a graph. If G is locally irregular then $\mathrm{I}(G) = 0$ and Algorithm 1 answers correctly (in line 2). Assume that G is not locally irregular; then there exist an edge $vu \in E$ such that $d_G(v) = d_G(v)$. Now, let S be an $ir^*(G)$. It follows from Observation 5 that S must include at least one vertex $w \in N_G[\{v, u\}]$. Since Algorithm 1 considers all the vertices in $N_G[\{v, u\}]$, at some point it also considers the vertex $w \in S \cap N_G[\{v, u\}]$. Now, observe that for any $x \in S$, the set $S_x = S \setminus \{x\}$ is an $ir^*(G_x)$, where $G_x = G[V \setminus \{x\}]$. Furthermore, by Lemma 21, we have $\mathrm{I}(G_x) \leq k - 1 = k_0$ iff $\mathrm{I}(G) \leq k$. By the induction hypothesis, we know that the algorithm answers correctly for all the instances (G_x, k_0). Thus, if $\mathrm{I}(G) \leq k = k_0 + 1$, there must exist one instance (G_w, k_0), where $w \in S \cap N_G[\{v, u\}]$, for which the Algorithm 1 returns yes. Therefore the algorithm answers for $(G, k_0 + 1)$ correctly. Finally, this process request $n^{\mathcal{O}(1)}$ steps in order to check if the graph is locally irregular and $2\Delta f(k - 1, n - 1)$ steps (by induction hypothesis) in order to check if for any graph G_x we have $\mathrm{I}(G_x) \leq k - 1 = k_0$ (where $x \in N[\{u, v\}]$). So, the algorithm decides in $n^{\mathcal{O}(1)} + 2\Delta f(k - 1, n - 1) \leq n^{\mathcal{O}(1)} + 2\Delta k(2\Delta)^{k-1}(n - 1)^{\mathcal{O}(1)} \leq n^{\mathcal{O}(1)} + k(2\Delta)^k n^{\mathcal{O}(1)} \leq (k + 1)(2\Delta)^k n^{\mathcal{O}(1)}$ steps. Finally, note that $k \leq n - 1$, and the result follows. ◀

A.2 Proof of Theorem 14

Proof. As the techniques we are going to use are standard, we are sketching some of the introductory details. For more details on tree decompositions (definition and terminology) see [19]. We are going to perform dynamic programming on the nodes of the given nice tree decomposition (see [10] for the definition of a nice tree decomposition). For a node t of the given tree decomposition of G, we denote by B_t the bag of this node and by $B_t^{\downarrow}$ the set of vertices of the graph that appears in the bags of the nodes of the subtree with t as a root. Observe that $B_t \subseteq B_t^{\downarrow}$.

The idea behind our algorithm, is that for each node t we store all the sets $S \subseteq B_t^{\downarrow}$ such that S is an $ir(G, B_t^{\downarrow} \setminus B_t)$. We will also store the necessary "conditions" (explained more in what follows) such that if there exists a set S', where $S' \setminus S \subseteq V \setminus B_t^{\downarrow}$, that meets these conditions, then S' is an $ir(G, B_t^{\downarrow})$. Observe that if we manage to do such a thing for every node of the tree decomposition, then we can find $\mathrm{I}(G)$. To do so, it suffices to check the size of all the irregulators we stored for the root r of the tree decomposition, which also meet the conditions we have set. In that way, we can find a set S that is an $ir(G, B_r^{\downarrow} \setminus B_r)$, satisfies our conditions and is of minimum order, and since $B_r^{\downarrow} = V$, this set S is a minimum irregulator of G and $\mathrm{I}(G) = |S|$.

Let us now present the actual information we are keeping for each node. Assume that t is a node of the tree decomposition and $S \subseteq B_t^{\downarrow}$ is an irregulator of $B_t^{\downarrow} \setminus B_t$ in G, i.e., S is an $ir(G, B_t^{\downarrow} \setminus B_t)$. For this S we want to remember which vertices of B_t belong to S as well as the degrees of the vertices $v \in B_t \setminus S$ in $G[B_t^{\downarrow} \setminus B_t]$. This can be done by keeping a table D of size $tw + 1$ where, if $v \in B_t \setminus S$ we set $D(v) = d_{G[B_t^{\downarrow} \setminus B_t]}(v)$ and if $v \in B_t \cap S$ we set $D(v) = \emptyset$ (slightly abusing the notation, by $D(v)$ we mean the position in the table D that corresponds to the vertex v). Like we have already said, we are going to keep some additional information about the conditions that could allow these sets to be extended to irregulators of $B_t^{\downarrow}$ in G if we add vertices of $V \setminus B_t^{\downarrow}$. For that reason, we are also going to keep a table with the "*target degree*" of each vertex; in this table we assign to each vertex $v \in B_t \setminus S$ a degree d_v such that, if there exists S' where $S' \setminus S \subseteq V \setminus B_t^{\downarrow}$ and for all $v \in B_t \setminus S$ we have $d_{G[V \setminus S']}(v) = d_v$, then S is an $ir(G, B_t^{\downarrow})$. This can be done by keeping a table T of size $tw + 1$ where for each $v \in B_t \setminus S$ we set $T(v) = i$, where i is the target degree, and for each $v \in B_t \cap S$ we set $T(v) = \emptyset$. Such tables T will be called *valid* for S in B_t. Finally, we are going to keep the set $X = S \cap B_t$ and the value $min = |S|$. Note that the set X does not gives us any extra information, but we keep it as it will be useful to refer to it directly.

To sum up, for each node t of the tree decomposition of G, we keep a set of quadruples (X, D, T, min), each quadruple corresponding to a valid combination of a set S that is an $ir(G, B_t^{\downarrow} \setminus B_t)$ and the target degrees for the vertices of $B_t \setminus S$. Here it is important to say that when treating the node B_t, for every two quadruples (X_1, D_1, T_1, min_1) and (X_2, D_2, T_2, min_2) such that for all $v \in B_t$ we have that $D_1(v) = D_2(v)$ and $T_1(v) = T_2(v)$ (this indicates that $X_1 = X_2$ as well), then we are only going to keep the quadruple with the minimum value between min_1 and min_2 as we will prove that this is enough in order to find I(G).

▷ Claim 22. Assume that for a node t, we have two sets S_1 and S_2 that are both $ir(G, B_t^{\downarrow} \setminus B_t)$, and that T is a target table that is common to both of them. Furthermore, assume that $(X_1, D_1, T, |S_1|)$ and $(X_2, D_2, T, |S_2|)$ are the quadruples we have to store for S_1 and S_2 respectively (both respecting T), with $D_1(v) = D_2(v)$ for every $v \in B_t$. Then for any set $S \subseteq V \setminus B_t^{\downarrow}$ such that $d_{G[V \setminus (S_1 \cup S)]}(v) = T(v)$ for all $v \in B_t$, we also have that $d_{G[V \setminus (S_2 \cup S)]}(v) = T(v)$ for all $v \in B_t$.

Proof. Assume that we have such an S for S_1, let v be a vertex in B_t and $H = G[v \cup \big((V \setminus B_t^{\downarrow}) \setminus S\big)]$ (observe that H does not depend on S_1 or S_2). Since $d_{G[V \setminus (S_1 \cup S)]}(v) = T(v)$, we know that in the graph H, v has exactly $T(v) - D_1(v)$ neighbours (as $D_1(v) = d_{G[B_t^{\downarrow} \setminus S_1)]}(v)$). Now, since $D_1(v) = D_2(v) = d_{G[B_t^{\downarrow} \setminus S_2]}(v)$ we have that $d_{G[V \setminus S_2 \cup S]}(v) = T(v)$. Therefore, the claim holds. ◁

Simply put, Claim 22 states that for any two quadruples $Q_1 = (X, D, T, min_1)$ and $Q_2 = (X, D, T, min_2)$, any extension S of S_1 is also an extension of S_2 (where S_1 and S_2 are the two sets that correspond to Q_1 and Q_2 respectively). Therefore, in order to find the minimum solution, it is sufficient to keep the quadruple that has the minimum value between min_1 and min_2.

Now we are going to explain how we create all the quadruples (X, D, T, min) for each type of node in the tree decomposition. First we have to deal with the Leaf Nodes. For a Leaf node t we know that $B_t = B_t^{\downarrow} = \emptyset$. Therefore, we have only one quadruple (X, D, T, min), where the size of both D and T is zero (so we do not need to keep any information in them), $S = \emptyset$ and $min = |S| = 0$.

Now let t be an Introduce node; assume that we have all the quadruples (X, D, T, min) for its child c and let v be the introduced vertex. By construction, we know that v is introduced in B_t and thus it has no neighbours in $B_t^{\downarrow} \setminus B_t$. It follows that if $S \subseteq B_c^{\downarrow}$ is an irregulator for $B_c^{\downarrow} \setminus B_c$, then both S and $S \cup \{v\}$ are irregulators for $B_t^{\downarrow} \setminus B_t$ in G. Furthermore, there is no set $S \subseteq B_t^{\downarrow} \setminus \{v\}$ that is an irregulator of $B_t^{\downarrow} \setminus B_t$ and is not an irregulator of $B_c^{\downarrow} \setminus B_c$. So, we only need to consider two cases for the quadruples we have to store for c; if v belongs in the under-construction irregulator of $B_t^{\downarrow} \setminus B_t$ in G or not.

Case 1. (v is in the irregulator): Observe that for any S that is an $ir(G, B_c^{\downarrow} \setminus B_c)$, which is stored in the quadruples of B_c, for every $u \in B_c \setminus S$, we have that $d_{G[B_c^{\downarrow} \setminus S]}(u) = d_{G[B_t^{\downarrow} \setminus (S \cup \{v\})]}(u)$. Moreover, for any target table T which is valid for S in c, the target table T' is valid for $S \cup \{v\}$ in t, where T' is almost the same as T, the only difference being that T' also contains the information about v, *i.e*, $T'(v) = \emptyset$. So, for each quadruple (X, D, T, min) in c, we need to create one quadruple $(X \cup \{v\}, D', T', min + 1)$ for t, where D' is the almost the same as D, except that it also contains the information about v, i. e., $D'(v) = \emptyset$.

Case 2. (v is not in the irregulator): Let $q = (X, D, T, min)$ be a stored quadruple of c and S be the corresponding $ir(G, B_c^{\downarrow} \setminus B_c)$. We will first explain how to construct D' of t, based on q. Observe that the only change between $G[B_c^{\downarrow} \setminus S]$ and $G[B_t^{\downarrow} \setminus S]$, is that in the latter there exist some new edges from v to some of the vertices of B_c. Therefore, for each vertex $u \in B_c \setminus X$ we set $D'(u) = D(u) + 1$ if $u \in N[v]$ and $D'(u) = D(u)$ otherwise. Finally, for the introduced vertex v, we set $D'(v) = |N(v) \cap (B_c \setminus X)|$. We will now treat the target degrees for t. Observe that the target degrees for each vertex in $B_t \setminus \{v\}$ are the same as in T, since v only has edges incident to vertices in B_t. Now, we only need to decide which are the valid targets for v. Since $d_{G[B_t^{\downarrow} \setminus S]}(v) = D'(v)$, we know that for every target t', we have that $D'(v) \le t' \le \Delta$. Furthermore, we can not have the target degrees of v to be the same as the targets of one of its neighbours in B_c (these values are stored in T), as, otherwise, any valid target table T' of t would lead to adjacent vertices in B_t having the same degree. Let $\{t_1, \dots, t_k\} \subset \{D(v), \dots, \Delta\}$ be an enumeration of all the valid targets for v (i.e. $t_i \neq T(u)$ for all $u \in N[v] \cap B_c \setminus X$). Then, for each quadruple (X, D, T, min) in c, and for each $i = 1, \dots, k$, we need to create the quadruple (X, D', T_i, min), such that $T_i(u) = T(u)$ for all $u \in B_c$ and $T_i(v) = t_i$. In total, we have $k \le \Delta$ such quadruples.

Now, let us explain how we deal with the Join nodes. Assume that t is a Join Node with c_1 and c_2 as its two children in the tree decomposition. Here, it is important to mention that $B_{c_1} = B_{c_2}$ and $(B_{c_1}^{\downarrow} \setminus B_{c_1}) \cap (B_{c_2}^{\downarrow} \setminus B_{c_2}) = \emptyset$. Assume that there exists an irregulator S of $B_t^{\downarrow} \setminus B_t$ in G, a valid target table T of S, and let (X, D, T, min) be the quadruple we need to store in t for this pair (S, T). Observe that this pair (S, T) is valid for both c_1 and c_2, so we must already have stored at least one quadruple in each node. Let $X \subseteq B_t$ and a target table T such that (X, D_1, T, min_1) and (X, D_2, T, min_2) are stored for c_1 and c_2 respectively. We create the quadruple (X, D, T, min) for t by setting $D(u) = D_1(u) + D_2(u) - d_{G[B_t \setminus X]}(u)$ for all $u \in B_t \setminus X$, $D(u) = \emptyset$ for all $u \in X$ and $min = min_1 + min_2 - |X|$. Observe that these are the correct values for the $D(u)$ and min, as otherwise we would count $d_{G[B_t \setminus X]}(u)$ and $|X|$ twice. Finally, we need to note that we do not store any quadruple (X, D, T, min) we create for the Join Note such that $D(u) > T(u)$ for a vertex $u \in B_t \setminus X$. This is because for such quadruples, the degree of vertex u will never be equal to any of the target degrees we have set, as it can only increase when we consider any of the ancestor (i. e. parent, grantparent etc.) nodes of t.

Finally, we need to treat the Forget nodes. Let t be a Forget node, c be the its child and v be the forgotten vertex. Assume that we have to store in t a quadruple (X, D, T, min). Then, since $X = B_t \cap S$ for an irregulator S of B_t in G, we know that in c we must have already stored a quadruple (X', D', T', min') such that, $X' = S \cap B_c$, $D'(u) = D(u)$ for all $u \in B_c$, $T'(u) = T(u)$ for all $u \in B_c$ and $min' = min$. Therefore, starting from the stored quadruples in c, we can create all the quadruples of t. For each quadruple (X', D', T', min') in c, we create at most one quadruple (X, D, T, min) for t by considering two cases; the forgotten vertex v_f belongs to X' or not.

Case 1. *(v belongs to X')*: then the quadruple (X, D, T, min) is almost the same as (X', D', T', min'), with the following differences: $X = X' \setminus \{v\}$, $min = min'$, $D(u) = D'(u)$ and $T(u) = T'(u)$ for all $u \in B_t$ and the tables D and T do not include any information for v as this vertex does not belong to B_t anymore.

Case 2. *(v does not belong to X')*: we will first check if $D'(v_f) = T'(v_f)$ or not. This is important because the degree of the v will never again be considered by our algorithm, and thus its degree will remain unchanged. So, if $D'(v_f) = T'(v_f)$, we create the quadruple (X, D, T, min) where $X = X'$, $min = min'$, $D(u) = D'(u)$ and $T(u) = T'(u)$ for all $u \in B_t$ and the tables D and T do not include any information for v.

For the running time, observe that the number of nodes of a nice tree decomposition is $\mathcal{O}(tw \cdot n)$ and all the other calculations are polynomial in $n + m$. Thus we only need to count the different quadruples in each node. Now, for each vertex v, we either include it in X or we have $\Delta + 1$ options for the value $D(u)$ and $\Delta + 1 - i$ for the value $T(u)$ if $D(u) = i$. Also, for sufficiently large Δ, we have that $1 + \sum_{i=0}^{\Delta}(\Delta + 1 - i) < \Delta^2$. Furthermore, the set X and the value min do not increase the number of quadruples because $X = \{u \mid D(u) = \emptyset\}$ and from all quadruples (X, D_1, T_1, min_1), (X, D_2, T_2, min_2) such that $D_1(u) = D_2(u)$ and $T_1(u) = T_2(u)$ for all $u \in B_t$, we only keep one of them (by Claim 22).

In total, the number of different quadruples in each node is Δ^{2tw}, and therefore the algorithm decides in $\Delta^{2tw} n^{\mathcal{O}(1)}$ time. ◀