Journal of Computer and System Sciences 132 (2023) 68–88

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

journal homepage: www.elsevier.com/locate/jcss

Building large k-cores from sparse graphs [☆], [☆☆]

Fedor V. Fomin [a,*], Danil Sagunov [b], Kirill Simonov [c]

[a] Department of Informatics, University of Bergen, Thormøhlens Gate 55, 5008 Bergen, Norway
[b] St. Petersburg Department of V.A. Steklov Institute of Mathematics, Fontanka River Embankment 27, 191011 Saint Petersburg, Russia
[c] Hasso Plattner Institute, University of Potsdam, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

ARTICLE INFO

Article history:
Received 8 November 2021
Received in revised form 10 October 2022
Accepted 20 October 2022
Available online 8 November 2022

Keywords:
Parameterized complexity
k-Core
Vertex cover
Treewidth

ABSTRACT

A k-core of a graph G is the maximal induced subgraph in which every vertex has degree at least k. In the EDGE k-CORE optimization problem, we are given a graph G and integers k, b and p. The task is to ensure that the k-core of G has at least p vertices, by adding at most b edges. While EDGE k-CORE is known to be computationally hard in general, we show that there are efficient algorithms when the k-core has to be constructed from a sparse graph with some structural properties. Our results are as follows.

- When the input graph is a forest, EDGE k-CORE is solvable in polynomial time.
- EDGE k-CORE is fixed-parameter tractable (FPT) when parameterized by the minimum size of a vertex cover in the input graph.
- EDGE k-CORE is FPT when parameterized by the treewidth of the graph plus k.

© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

The k-core in an undirected graph G is the maximal induced subgraph of G in which all vertices have degree at least k. This concept has been applied in various areas including social networks [2–4], protein function prediction [5], hierarchical structure analysis [6], graph visualization [7], and network clustering and connectivity [8,9].

In online social networks users tend to contribute content only when a certain amount of their friends do the same [10], or in other words, when the formed community is a k-core for some threshold parameter k. Interestingly, losing even a small amount of users or links can bring to the cascade of iterated withdrawals. A classical example of such phenomena is the example of Schelling from [11]: Consider a cycle on n vertices, which is a 2-core with n vertices. Missing just one edge from this graph turns it into a path and triggers withdrawals that result in dismounting of the whole network. On the other hand, adding a small number of extra links can create a large k-core and thus prevent users from withdrawing. We consider the following mathematical model for this problem. For a given network, the assumption is that a user leaves the network when less than k of his/her friends remain within it. We would like to prevent unraveling of the network, so that at least

[☆] The short version of this paper previously appeared on MFCS 2020 [1].
[☆☆] This work is supported by the Research Council of Norway via the project "MULTIVAL" (grant no. 263317). Kirill Simonov acknowledges support by DFG Research Group ADYN under grant DFG 411362735.
[*] Corresponding author.
E-mail addresses: Fedor.Fomin@uib.no (F.V. Fomin), danilka.pro@gmail.com (D. Sagunov), kirillsimonov@gmail.com (K. Simonov).

https://doi.org/10.1016/j.jcss.2022.10.002
0022-0000/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

p users remain engaged in it. To achieve this, we are given a budget to establish at most b new connections between the users of the network. More precisely, the problem is stated as follows.

Edge k-Core

Input: A simple undirected graph G and integers b, k, and p.

Task: Decide whether there exists $B \subseteq \binom{V(G)}{2} \setminus E(G)$ of size at most b such that the k-core of the graph $(V(G), E(G) \cup B)$ is of size at least p.

The Edge k-Core problem was introduced by Chitnis and Talmon in [12] as a model of preventing unraveling in networks. For instance, in a P2P network, any user benefiting from the network should be linked to at least k other users exchanging resources. In this scenario the Edge k-Core model could be used to find extra connections between users to provide a better service for a larger number of users [12,13]. Other potential applications of Edge k-Core in real-life networks include friend recommendation in social networks, connection construction in telecom networks, etc. We find the Edge k-Core problem to be interesting from the theoretical perspective too: it has strong links to the well-studied family of problems, where one seeks a modification of a graph satisfying certain conditions on the degrees of the vertices, see [14] for further references. Our interest in the study of the problem is of a theoretical nature.

The k-core in a graph can be found by a simple "shaving" procedure: If a graph contains a vertex of degree less than k, then this vertex cannot be in its k-core and thus can be safely removed. Apparently, solving Edge k-Core is more challenging. In particular, Chitnis and Talmon in [12] proved that Edge k-Core is NP-complete even for $k = 3$ and when the input graph G is 2-degenerate.[1] Moreover, the problem is W[1]-hard being parameterized by $k + b + p$. On the other hand, they show that if the treewidth of the graph G is tw, then the problem is solvable in time $(k + \mathrm{tw})^{\mathcal{O}(\mathrm{tw}+b)} \cdot n^{\mathcal{O}(1)}$ and hence is fixed-parameter tractable (FPT) parameterized by $k + \mathrm{tw} + b$. These results of Chitnis and Talmon are the departure point for our study.

Our results. We study the algorithmic complexity of Edge k-Core on three families of sparse graphs: forests, graphs with bounded vertex cover number and graphs of bounded treewidth. Each of our algorithms is based on one of the common algorithmic paradigms: dynamic programming for forests and treewidth, and ILP for vertex cover. The interesting part here is that in each of the cases, the successful application of an algorithmic paradigm crucially depends on a new combinatorial result. We show the following.

Growing from forest. We prove (Theorem 5) that Edge k-Core is solvable in time $\mathcal{O}(k \cdot |V(G)|^2)$, when the input graph G is a forest. The algorithm is based on a dynamic programming over subtrees. The crucial part of the work is to make this algorithm run in polynomial time. For that we need a new graph-theoretical result, Theorem 4. The theorem states that for any integer k, a forest F on at least $k + 1$ vertices can be completed into a graph of minimum degree k by adding at most

$$\left\lceil \frac{1}{2} \sum_{v \in V(F)} \max\{0, k - \deg(v)\} \right\rceil$$

edges. Moreover, this bound is tight, any forest requires such amount of edge additions to grow into a k-core. The proof of Theorem 4 is non-trivial and exploits an interesting connection between the cores in a graph and sufficient conditions on the existence of a large matching in a graph. Here the recent combinatorial theorem of Henning and Yeo [15] on matchings in graphs of bounded degrees becomes handy.

Bounded vertex cover. We prove that the problem is FPT parameterized by the minimum size of a vertex cover in a graph. More precisely, in Theorem 14, we give an algorithm of running time $2^{\mathcal{O}(\mathrm{vc} \cdot 3^{\mathrm{vc}})} \cdot n^{\mathcal{O}(1)}$, where vc is the vertex cover number of the input graph. Let us note that every graph is vc-degenerate. We solve the problem by reducing it to an integer linear program (ILP), whose number of variables is bounded by some function of vc. This allows to apply Lenstra's algorithm [16], see also [17,18], to solve Edge k-Core. Nowadays ILP is a commonly used tool for designing parameterized algorithms, see e.g. [19, Chapter 6]. However, just like in the case of forests, the application of an algorithmic paradigm is not direct. In order to encode the problem as ILP with the required number of variables, we need a new combinatorial result (Lemma 13) about degree sequences of a graph. Lemma 13 is essentially a characterization of graphic sequences, but stated in terms of the frequences of the degrees, in contrast to the classical Erdős-Gallai theorem [20] about graphic sequences.

From the viewpoint of parameterized complexity, the next natural question is whether Edge k-Core parameterized by vc admits a polynomial *kernel*, that is, a polynomial-time preprocessing algorithm that compresses a given instance into an equivalent one of size $(\mathrm{vc})^{\mathcal{O}(1)}$. While many graph-theoretic problems have polynomial kernels parameterized by vc [21], it turns out that Edge k-Core remains hard in this sense. We complement the FPT algorithm with a hardness result excluding the existence of a polynomial kernel, under a standard complexity assumption.

Bounded treewidth. Chitnis and Talmon in [12] have shown that Edge k-Core is FPT parameterized by $\mathrm{tw} + k + b$, where tw is the treewidth of the input graph. Even in the case when the treewidth and k are constants, this does not mean that

[1] Recall that a graph is d-degenerate if its every induced subgraph contains a vertex of degree at most d. Thus the d-core is the maximum subgraph which is not $d - 1$ degenerate.

the problem is solvable in polynomial time. We enhance this result by proving that Edge k-Core is FPT parameterized by $\text{tw} + k$. As the algorithm of Chitnis and Talmon in [12], our algorithm is a dynamic programming on graphs of bounded treewidth, but again, in order to make it work, we need a new combinatorial result (Theorem 20). When the budget b is small (of order k^3), the algorithm of Chitnis and Talmon suffices. When the budget b is large, we are able to approach the problem in an interesting new way. Here Theorem 20 provides us with a criterion of how a subset of vertices can be turned into a k-core "optimally". Specifically, if a graph "lacks" sufficiently many edges (polynomially many in k), then it can be completed to a k-core without adding edges that "waste" the degree, i.e., are incident to a vertex that already has a degree of at least k. This key insight allows us to show that the problem is FPT parameterized by $\text{tw} + k$, since by Theorem 20 the task reduces to finding a subgraph that "lacks" the least amount of edges solely in terms of vertex degrees, and that can be done by dynamic programming over the tree decomposition.

Related work. The usability of k-cores in the study of network unraveling phenomena was popularized by the influential paper of Bhawalkar et al. [2] who suggested the model of forcing a limited number of users of a network to stay in order to maximize the size of the k-core. The same problem was further studied in [22], where new computational results were obtained and some results of [2] were strengthened. Also, Chitnis, Fomin and Golovach studied this problem applied to networks where the underlying graph is directed [23]. Heuristic algorithms for this problem are discussed in [24].

Edge k-Core was introduced in [12], where also a number of complexity and algorithmic results about the problem were established. Zhou et al. [13] provide some non-approximability results for Edge k-Core as well as some heuristics. The work by Zhang et al. [25] is devoted to the "dual" problem of disengaging a limited number of users from a network in order to minimize its k-core size. Another work in this context is due to Luo, Molter and Suchy [26].

More generally, Edge k-Core fits into a large class of edge modification problems, where one is seeking for an optimum modification to some desired graph property [14]. In particular, a significant part of literature in parameterized complexity is devoted to related problems of graph modification to graphs with some vertex degree properties like being regular, Euler, or to some degree sequence [27–30].

2. Preliminaries

All graphs considered in this paper are simple undirected graphs. We use standard graph notation and terminology, following the book of Diestel [31]. We write $G + F$ to denote the simple graph obtained by adding the edges from $F \subseteq \binom{V(G)}{2} \setminus E(G)$ to a graph G. If not specified otherwise, we use n to denote the number of vertices of the graph G in an input instance of Edge k-Core. For two sets A and B we write $A \sqcup B$ to denote the disjoint union of A and B. That is, $A \sqcup B = A \cup B$, and A and B are required to be disjoint.

Throughout this paper, we use the following terms. In the following definitions, we assume that k is fixed.

Definition 1 *(Deficiency).* For a graph G and a vertex $v \in V(G)$, let $\text{df}_G(v) = \max\{0, k - \deg_G(v)\}$ denote the *deficiency* of v in G. We denote the *total deficiency* in G by $\text{df}(G) = \sum_{v \in V(G)} \text{df}_G(v)$.

Note that the addition of an edge between two vertices of G can decrease $\text{df}(G)$ by at most two. It also does not make any sense to add edges that do not decrease deficiency if we aim to complete G to a graph of minimum degree k. We distinguish added edges by whether they decrease deficiency by two or one.

Definition 2 *(Good/bad edges).* For nonadjacent vertices $u, v \in V(G)$ a new added edge uv is *good* if both $\text{df}_G(u) > 0$ and $\text{df}_G(v) > 0$. If $\text{df}_G(u) = 0$ and $\text{df}_G(v) > 0$, then uv is *bad*.

Thus adding a good edge decreases the total deficiency by 2 and adding a bad one by 1.

Definition 3 *(A k-core graph).* We say that a graph G is *a k-core* if G is the k-core of itself. We also say that a vertex set H in G *induces a k-core* in G if $G[H]$ is a k-core.

Note that whenever there is a vertex set H of size at least p which induces a k-core in G, the k-core of G has also size at least p, since it is the unique maximal induced subgraph of G which is a k-core. We often use this simple observation throughout the paper whenever we show that the k-core is large by presenting a large vertex set which induces a k-core.

3. Growing from forest

In this section we present our polynomial time algorithm for Edge k-Core on forests and the underlying graph-theoretical result.

The algorithm itself is a dynamic programming over subtrees. Normally, an algorithm like this would go from leaves to larger and larger subtrees, storing for every subtree a list of possible configurations a solution could induce on this subtree. In the Edge k-Core problem, naturally we want to store information about edges added inside the subtree and vertices from the subtree which we may later connect to something outside.

Naively, this would take exponential space, as it seems we have to store at least the degrees of the selected vertices in the subtree. However, the following theorem, which is the central technical result of this section, helps greatly.

Theorem 4. *For any integer k, any forest T on at least $k+1$ vertices can be completed to a graph of minimum degree k by adding at most*

$$\left\lceil \frac{1}{2} \sum_{v \in V(T)} \max\{0, k - \deg(v)\} \right\rceil$$

edges, and this cannot be done with less edge additions. Moreover, in the case $k \geq 4$, it can be done in a way that the added edges form a connected graph on the vertices they cover.

For our algorithm, Theorem 4 means that whenever we fix the subset of vertices H, we have to add exactly $\lceil \mathrm{df}(T[H])/2 \rceil$ edges in order to induce a k-core on H. Thus it is enough to find a subset of vertices H of size at least p with the smallest possible $\mathrm{df}(T[H])$. This objective turns out to be simple enough for the bottom-up dynamic programming. Namely, for a subtree T_v rooted at v, it is enough to store the size of $H \cap T_v$, the total deficiency of these vertices, whether v is in H and how many neighbors in $H \cap T_v$ it has. Since v separates T_v from the rest of the tree, the deficiency of other vertices in $H \cap T_v$ is unchanged no matter how H looks like in the rest of the tree.

The discussion above ultimately leads to a polynomial time algorithm, stated formally in the next theorem.

Theorem 5. Edge k-Core *is solvable in time $\mathcal{O}(kn^2)$ on the class of forests.*

Proof. Let (G, b, k, p) be the given instance of Edge k-Core, and G be a forest. Consider a subset $H \subseteq V(G)$ of the vertices of G. Since $G[H]$ is a forest, by Theorem 4, one needs $\lceil \frac{1}{2} \mathrm{df}(G[H]) \rceil$ edge additions to make H a (subset of a) k-core in G. Thus, to solve the given instance, it is enough to check whether there is a subset $H \subseteq V(G)$ with $|H| \geq p$, such that $2b \geq \mathrm{df}(G[H])$. Hence, it is enough to find H with the smallest value of $\mathrm{df}(G[H])$. In the rest of the proof, we show how to find such H in polynomial time using dynamic programming. The algorithm itself is in fact a special case of our treewidth dynamic programming, presented in Lemma 22 of Section 5. However, we describe the algorithm for forests in full detail here for the sake of completeness of this section.

To simplify our task, let us first make G connected. For that, introduce a new vertex r to G, and connect this vertex with each connected component of G by a single edge arbitrarily, even if G consists of a single connected component. We thus obtained a tree T that differs from G only in the newly-introduced vertex r. For each $H \subseteq V(G)$, $G[H] = T[H]$. Hence, we are now looking for a subgraph in T on at least p vertices, such that this subgraph does not contain r, and we want its total deficiency to be minimum possible.

Make T rooted in r. For $v_i \in V(T)$, by T_{v_i} we denote the subtree of the vertex v_i in T. Let t_i be the number of child vertices of v_i in T. By $c_{i,1}, c_{i,2}, \dots, c_{i,t_i}$ denote the children of v_i in T in arbitrary order. If v_i is a leaf vertex, $t_i = 0$. For each $j \in \{0, 1, \dots, t_i\}$ by $T^j_{v_i}$ denote the subtree of v_i in T, but including only subtrees of the first j of its children. That is,

$$T^j_{v_i} = T_{v_i}[\{v_i\} \sqcup V(T_{c_{i,1}}) \sqcup V(T_{c_{i,2}}) \sqcup \dots \sqcup V(T_{c_{i,j}})].$$

Clearly, $T_{v_i} = T^{t_i}_{v_i}$.

Now for each $v_i \in V(T)$, each $j \in \{0, \dots, t_i\}$ and each $s \in \{0, 1, \dots, |V(T^j_{v_i})| - 1\}$, let

$$OPT^j_0(v_i, s) = \min \left\{ \mathrm{df}(T^j_{v_i}[S]) \;\middle|\; \begin{array}{c} S \subset V(T^j_{v_i}), \\ |S| = s, \\ v_i \notin S \end{array} \right\}.$$

In other words, $OPT^j_0(v_i, s)$ equals the minimum deficiency of a subgraph of $T^j_{v_i}$ on exactly s vertices, such that it does not contain v_i. Also denote $OPT_0(v_i, s) = OPT^{t_i}_0(v_i, s)$.

On the other hand, for each $v_i \in V(T)$, each $j \in \{0, 1, \dots, t_i\}$ each $s \in [|V(T^j_{v_i})|]$ and each $d \in \{0, 1, \dots, k\}$, define

$$OPT^j_1(v_i, s, d) = \min \left\{ \mathrm{df}(T^j_{v_i}[S]) \;\middle|\; \begin{array}{c} S \subseteq V(T^j_{v_i}), \\ |S| = s, \\ v_i \in S, \\ \mathrm{df}_{T^j_{v_i}[S]}(v_i) = d \end{array} \right\}.$$

Thus, $OPT^j_1(v, s, d)$ equals the minimum deficiency of a subgraph of $T^j_{v_i}$ on s vertices, including v_i, such that the deficiency of v_i in this subgraph equals d. For some choice of v_i, j, s, d, there may be no corresponding subgraphs. In such cases, we put $OPT^j_1(v_i, s, d) = \infty$. Also denote $OPT_1(v_i, s, d) = OPT^{t_i}_1(v_i, s, d)$.

Finally, for each $v \in V(T)$ and each $s \in \{0, 1, \ldots, |V(T_v)|\}$, let

$$OPT(v_i, s) = \min\left\{OPT_0(v_i, s),\ \min_{d=0}^{k} OPT_1(v_i, s, d)\right\}$$

denote the minimum deficiency of a subgraph of T_{v_i} on s vertices.

Clearly, the minimum possible deficiency we are looking for equals

$$\min_{i=p}^{|V(G)|} OPT_0(r, i),$$

and it is enough to compare this value with $2b$ to solve the initial instance of Edge k-Core. We now show how we compute the values of OPT. We do this in a bottom-up manner, starting from the leaf vertices of T.

Let $v_i \in V(T)$ be a vertex in T. Since $T^0_{v_i}$ consists of a single vertex, the only choice of s for $OPT^0_0(v_i, s)$ is $s = 0$, and $OPT^0_0(v_i, 0) = 0$. For $OPT^0_1(v_i, s, d)$, the only choice is $s = 1$ and $d = k$, and $OPT^0_1(v_i, 1, k) = k$, as $\text{df}(T^0_{v_i}) = k$. If v_i is a leaf vertex, then computations for v_i are finished, as $OPT_0(v_i, s) = OPT^0_0(v_i, s)$ and $OPT_1(v_i, s, d) = OPT^0_1(v_i, s, d)$.

Otherwise, $t_i > 0$, and we can assume that all values of OPT for the children of v_i are already computed. Consider computing $OPT^j_0(v_i, s)$ for $j > 0$. Take $S \subset V(T^j_{v_i})$, $v_i \notin S$. Since subtrees of the children of v_i are connected only through v_i, it is true that

$$\text{df}(T^j_{v_i}[S]) = \text{df}(T^{j-1}_{v_i}[S \cap V(T^{j-1}_{v_i})]) + \text{df}(T_{c_{i,j}}[S \cap V(T_{c_{i,j}})]). \tag{1}$$

In some sense, we can minimize total deficiencies of $T^{j-1}_{v_i}$ and $T_{c_{i,j}}$ separately. We obtain

$$OPT^j_0(v_i, s) = \min_{s_i=0}^{\min\{s, |V(T_{c_{i,j}})|\}} \left(OPT^{j-1}_0(v_i, s - s_i) + OPT(c_{i,j}, s_i)\right).$$

Thus, $OPT^j_0(v_i, s)$ is computed in $\mathcal{O}(|T_{c_{i,j}}|)$ time. We compute these values in increasing order of j. Since there are $|T^j_{v_i}|$ choices of s for a fixed j, computing all values of OPT_0 for v_i in total takes $\mathcal{O}(|T^0_{v_i}| \cdot |T_{c_{i,1}}| + |T^1_{v_i}| \cdot |T_{c_{i,2}}| + \ldots + |T^{t_i-1}_{v_i}| \cdot |T_{c_{i,t_i}}|)$ time.

We now turn onto computing $OPT^j_1(v_i, s, d)$ for $j > 0$. Take $S \subseteq V(T^j_{v_i})$, $v_i \in S$. Denote $S_0 = S \cap V(T^{j-1}_{v_i})$ and $S_1 = S \cap V(T_{c_{i,j}})$, so $S = S_0 \sqcup S_1$. The vertex v_i is now included in the subgraph $T^j_{v_i}[S]$, and it may now influence the deficiency of its child $c_{i,j}$. Thus, equation (1) is not quite correct in this case. If $v_i \in S$ and $\text{df}_{T_{c_{i,j}}[S_1]}(c_{i,j}) > 0$, then in $T^j_{v_i}$ the deficiency of $c_{i,j}$ decreases by one because of the edge $v_i c_{i,j}$. If $\text{df}_{T^{j-1}_{v_i}[S_0]}(v_i) > 0$, then the deficiency of v_i in $T^j_{v_i}$ also decreases by one. Thus, if $c_{i,j} \in S$,

$$\text{df}(T^j_{v_i}[S]) = \text{df}(T^{j-1}_{v_i}[S_0]) - \min(1, \text{df}_{T^{j-1}_{v_i}[S_0]}(v_i)) + \text{df}(T_{c_{i,j}}[S_1]) - \min(1, \text{df}_{T_{c_{i,j}}[S_1]}(c_{i,j})), \tag{2}$$

and if $v_i \notin S$, equation (1) holds true. Denote by

$$JOIN(c_{i,j}, s_i) = \min\left\{OPT_1(c_{i,j}, s_i, 0), \min_{d_i=1}^{k}\left(OPT_1(c_{i,j}, s_i, d_i) - 1\right)\right\}$$

the minimum total deficiency of a subgraph of $T_{c_{i,j}}$ on s_i vertices including $c_{i,j}$, but with the deficiency of $c_{i,j}$ decreased by one, if it is non-zero. This corresponds to the second half of the right part of equation (2). Each value $JOIN(c_{i,j}, s_i)$ can be computed in $\mathcal{O}(k)$ time, and there are $|T_{c_{i,j}}|$ choices of s_i for any $j \in [t_i]$. Hence, all values of $JOIN$ for the children of v_i together can be computed in total $\mathcal{O}(|T_{c_{i,1}}| \cdot k + |T_{c_{i,2}}| \cdot k + \ldots + |T_{c_{i,t_i}}| \cdot k) = \mathcal{O}(|T_{v_i}| \cdot k)$ running time. We then obtain

$$OPT^j_1(v_i, s, d) = \min_{s_i=0}^{\min\{s, |V(T_{c_{i,j}})|\}} \left\{ \begin{array}{c} OPT^{j-1}_1(v_i, s - s_i, d) + OPT_0(c_{i,j}, s_i), \\ OPT^{j-1}_1(v_i, s - s_i, d + 1) + JOIN(c_{i,j}, s_i) \end{array} \right\}, \tag{3}$$

for any $s \in [|V(T^j_{v_i})|]$ and any $d < k$. The first argument of min in equation (3) corresponds to $v_i \notin S$ and equation (1), and the second argument corresponds to $v_i \in S$ and equation (2). If $d = k$, i.e., the deficiency of v_i in S equals k, then necessarily $v_i \notin S$. Thus, in the case $d = k$, we compute $OPT^j_1(v_i, s, d)$ according to equation (3), but without the second argument of min. This finishes the description of formulas for computing the values of OPT_1. Each separate value of $OPT^j_1(v_i, s, d)$ is computed in $\mathcal{O}(|T_{c_{i,j}}|)$ time. There are $|T^j_{v_i}| \cdot (k+1)$ choices of (s,d), so computing all values of OPT^j_1 for v_i together takes $\mathcal{O}(k \cdot \sum_{j=0}^{t_i} |T^j_{v_i}| \cdot |T_{c_{i,j}}|)$ running time. Note that this also covers the running time needed for computing the values of $JOIN$ for all children of v_i.

It is left to show that the total running time of the algorithm is $\mathcal{O}(k \cdot |V(T)|^2) = \mathcal{O}(k \cdot |V(G)|^2)$. Computing the values of OPT_0 and OPT_1 for a fixed vertex $v_i \in T$ takes $\mathcal{O}(k \cdot \sum_{j=0}^{t_i} |T^j_{v_i}| \cdot |T_{c_{i,j}}|)$ running time. We need to show that

$$\sum_{v_i \in V(T)} \sum_{j=0}^{t_i} |T^j_{v_i}| \cdot |T_{c_{i,j}}| \le |V(T)|^2,$$

which is equivalent to

$$\sum_{v_i \in V(T)} \sum_{j=0}^{t_i} |T^j_{v_i} \times T_{c_{i,j}}| \le |V(T) \times V(T)|.$$

Note that all sets $T^j_{v_i} \times T_{c_{i,j}}$ are pairwise-disjoint, and each of them is a subset of $V(T) \times V(T)$. Hence, the last inequality is essentially true. Thus, the dynamic programming itself is done in $\mathcal{O}(k \cdot |V(G)|^2)$ running time by the algorithm. Any other part of the algorithm, including constructing and rooting T and finding and comparing $\min_{i=p}^{|V(G)|} OPT_0(r,i)$ with $2b$, takes $\mathcal{O}(|V(G)|)$ time, which is covered by $\mathcal{O}(k \cdot |V(G)|^2)$. This finishes the proof. □

For the remaining part of this section we focus on the proof of Theorem 4.

Proof of Theorem 4. The theorem says that the completion of T to a graph of total deficiency 0 can be done using $\lceil \frac{1}{2}\operatorname{df}(T) \rceil$ edge additions. Note that this bound is tight because a single edge addition decreases the total deficiency by at most two. When $\operatorname{df}(T)$ is even, we have to prove that it is possible to complete T by adding only good edges. When $\operatorname{df}(T)$ is odd, we have to complete T to a graph of total deficiency 1 adding $\lfloor \frac{1}{2}\operatorname{df}(T) \rfloor$ good edges and then add one bad edge. Fixing deficiency 1 with one bad edge is always possible, since the only deficient vertex u has degree $k-1$ and so must have a non-neighbor. In the case $k \ge 4$ this can be also done in a way that connects u to the already added good edges. Thus, from now on, it suffices to prove that we can add $\lfloor \frac{1}{2}\operatorname{df}(T) \rfloor$ good edges, in a connected way for $k \ge 4$.

For $k=1$, vertices with non-zero deficiency are exactly the isolated vertices of T. In this case pairing isolated vertices arbitrarily provides the required $\lfloor \frac{1}{2}\operatorname{df}(T) \rfloor$ good edges.

For $k \ge 2$, it is sufficient to prove the theorem statement for the case when T is connected, i.e., T is a tree. If T is a forest consisting of at least two trees, one may reduce the number of trees in T. This can be done by picking two leaf vertices of distinct connected components in T and adding an edge between them. Clearly, such an edge addition is good since any leaf vertex has non-zero deficiency, and it reduces the number of connected components in T.

Moreover, for $k=2$, vertices with non-zero deficiency are exactly the leaves of T. Since T is a tree with at least three vertices, an edge connecting any two leaves can be added. Thus, as in the case of $k=1$, pairing the leaves arbitrarily suffices.

Now, for every integer $k \ge 3$, we prove Theorem 4 by induction on the number of vertices in the tree. The fact that the graph on the added edges must be connected in the case $k \ge 4$ will be useful for the induction.

Base case. Let T be a tree on $n = k+1$ vertices. The only way to complete T to a graph of minimum degree k is to turn it into a complete graph, i.e., add every possible missing edge between vertices in $V(T)$. Clearly, each edge addition in such a completion is good, thus the completion requires exactly $\frac{1}{2}\operatorname{df}(T)$ edge additions. Suppose there are two connected components formed by the added edges. Then T must contain all edges between these components, so it also contains a cycle, since each of the components has at least two vertices. Thus the connectivity condition must be satisfied.

Inductive step. Suppose that Theorem 4 holds for all trees on n vertices, and let T be a tree on $n+1$ vertices. We prove that Theorem 4 holds for T. Let v be a leaf of T and let $T' = T - v$ be the tree obtained by deleting v from T. By the induction hypothesis, T' can be completed to a graph of total deficiency ($\operatorname{df}(T') \bmod 2$) using $\lfloor \frac{1}{2}\operatorname{df}(T') \rfloor$ edge additions. Let A' be the graph on the deficient vertices of T' formed by the good edges added during the completion.

Our ultimate goal is to transform A' in such a way that it accounts for the new vertex v as well. We shall do this by first removing edges from A', and then adding good edges between vertices which are not yet adjacent. In the case $k \ge 4$, we must also end up with a connected graph on the added edges.

Briefly explained, our technique of adding and removing edges is as follows. Take an edge $st \in E(A')$, such that 1) $s \ne v$, $t \ne v$ and 2) sv and tv are not yet in the graph. Delete the edge st, and add both edges sv and tv. This operation preserves deficiencies of both s and t, while it decreases the deficiency of v by two. Note that s and t also remain connected through v. We can do the same with a matching instead of a single edge, thus we need a matching of size roughly $k/2$ to nullify the deficiency of v.

The rest of the proof is structured in two parts. First, we show that there is indeed a sufficiently large matching in A'. Second, we give a detailed description of how to reroute the edges of the matching to the new vertex v, and carefully verify the correctness of the procedure.

Finding a matching. We will need the following properties of A'.

$$\text{If } k \geq 4,\ A' \text{ is connected.} \tag{4}$$

The correctness of (4) follows from the induction hypothesis. Because each vertex in T' has deficiency at most $k-1$ and each edge addition is good, we have that

$$\Delta(A') \leq k-1. \tag{5}$$

Also

$$|E(A')| \geq \frac{n(k-2)+1}{2}, \tag{6}$$

since there must be at least $\frac{nk-1}{2}$ edges in the graph after the completion to deficiency $(\mathrm{df}(T') \bmod 2)$, and only $n-1$ of the edges are in T'.

$$|V(A')| \geq k. \textit{ If } n > k+1, \textit{ then } |V(A')| > k. \tag{7}$$

To prove (7), suppose that $|V(A')| \leq k-1$. By (5), $2 \cdot |E(A)| \leq \Delta(A') \cdot |V(A')| \leq (k-1)^2$. Then by (6), we have that either $(k-1)^2 \geq n(k-2)+1$, or $k^2-2k \geq n(k-2)$. Hence $k \geq n$, which is a contradiction. Therefore, $|V(A')| \geq k$.

Suppose now that $|V(A')| = k$, but $n \geq k+2$. Then either $k(k-1) \geq n(k-2)+1 \geq (k+2)(k-2)+1$, or $k^2-k \geq k^2-3$, so $k \leq 3$. Thus, $k=3$ and $n=k+2=5$ (if $n > k+2$, k should be strictly less than 3). Since $|V(A')| = k = n-2$, T' should have two vertices of degree at least three. But then T' should contain at least five edges, but it only contains four. A contradiction.

We now use these properties of A' to show that there is a large matching in A'. For lower bounds on the size of a maximum matching we rely on the recent work of Henning and Yeo [15].

Proposition 6 *([15]). For any integer $t \geq 3$, any connected graph G with $|V(G)| = n$, $|E(G)| = m$ and $\Delta(G) \leq t$, contains a matching of size at least*

$$\left(\frac{t-1}{t(t^2-3)}\right)n + \left(\frac{t^2-t-2}{t(t^2-3)}\right)m - \frac{t-1}{t(t^2-3)}, \textit{ if } t \textit{ is odd,}$$

or at least

$$\frac{n}{t(t+1)} + \frac{m}{t+1} - \frac{1}{t}, \textit{ if } t \textit{ is even.}$$

We shall use Proposition 6 to show that A' contains a matching of size roughly $\frac{k}{2}$, as stated in the following claim.

Claim 7. *When k is odd and $n = k+1$, A' has a matching of size at least $\frac{k-1}{2}$. Otherwise, A' has a matching of size at least $\lceil \frac{k}{2} \rceil$.*

Proof. Consider the case when k is even. We apply Proposition 6 with $t = k-1$ and A', and use properties (4) and (5). Since the lower bound from the theorem statement is rounded up, we need to show that

$$\left(\frac{t-1}{t(t^2-3)}\right)|V(A')| + \left(\frac{t^2-t-2}{t(t^2-3)}\right)|E(A')| - \frac{t-1}{t(t^2-3)} > \frac{k}{2} - 1.$$

By properties (6) and (7), and by replacing k with $t+1$, it is sufficient to show that

$$\frac{(t-1)(t+1)}{t(t^2-3)} + \left(\frac{t^2-t-2}{t(t^2-3)}\right) \cdot \left(\frac{n(t-1)+1}{2}\right) - \frac{t-1}{t(t^2-3)} > \frac{t+1}{2} - 1, \tag{8}$$

for any odd $t \geq 3$ and any $n \geq k+1 = t+2$. After multiplying both parts of (8) by $2t(t^2-3)$, we obtain

$$2(t-1)(t+1) + (t^2-t-2) \cdot (n(t-1)+1) - 2(t-1) > (t+1-2) \cdot t(t^2-3),$$

which is simplified to

$$n \cdot (t^3 - 2t^2 - t + 2) > t^4 - t^3 - 6t^2 + 6t + 2.$$

Since $n \geq t+2$, it is sufficient to show that

$$(t+2) \cdot (t^3 - 2t^2 - t + 2) > t^4 - t^3 - 6t^2 + 6t + 2,$$

or that

$$t^3 + t^2 - 6t + 2 > 0.$$

Since $t > 0$, we weaken the last inequality to

$$t^3 - 6t > 0,$$

which holds true for $t > \sqrt{6}$. Thus, for $t \geq 3$, inequality (8) holds. We hereby have shown that when k is even, A' has a matching of size $\frac{k}{2}$.

Consider now the case of $k = 3$. We need to consider this case separately since $t = k - 1 = 2$ does not follow from Proposition 6. When $k = 3$, we need a matching of size two in A'. If A' is not connected, it suffices to pick any two of its connected components and take an edge from each; note that by construction A' does not have isolated vertices. Otherwise, A' is connected and by (5) has maximum degree two. Hence, A' is either a simple cycle or a simple path. By property (6), A' is a cycle or a path consisting of at least $\lceil \frac{(k+1)(k-2)+1}{2} \rceil = \lceil \frac{4 \cdot 1+1}{2} \rceil = 3$ edges. Clearly, if A' is a path on at least three edges or a cycle on at least four edges, A' has a matching of size two. The only option left for A' is to be a cycle on three vertices and edges, so $|V(A')| = 3 = k$. By property (7), this is only possible if $n = k + 1$. Thus, if $n > k + 1$, A' has a matching of size at least two. If $n = k + 1$, A' is only guaranteed to have a matching of size $1 = \frac{k-1}{2}$.

It is left to consider the case of odd $k \geq 5$. Again, apply Proposition 6 to A' with even $t = k - 1$. We need to show that A' has a matching of size at least $\frac{k+1}{2} = \frac{t+2}{2}$, so it is sufficient to show that

$$\frac{|V(A')|}{t(t+1)} + \frac{|E(A')|}{t+1} - \frac{1}{t} > \frac{t+2}{2} - 1.$$

According to (6) and (7), it is enough to show that

$$\frac{t+1}{t(t+1)} + \frac{1}{t+1} \cdot \frac{1}{2} \cdot (n(t-1)+1) - \frac{1}{t} > \frac{t+2}{2} - 1,$$
$$\frac{(t-1)n+1}{2t+2} > \frac{t}{2}.$$

Multiplying both parts by $2t + 2$ and using $n \geq k + 2 = t + 3$, we obtain

$$(t-1)(t+3) + 1 > t(t+1),$$

or, equivalently, $t > 2$. Thus, when $k \geq 5$ is odd and $n \geq k + 2$, A' has a matching of size $\frac{k+1}{2}$.

When $n = k + 1$, we need a matching of size $\frac{k-1}{2} = \frac{t}{2}$, so we need to show that

$$\frac{|V(A')|}{t(t+1)} + \frac{|E(A')|}{t+1} - \frac{1}{t} > \frac{t}{2} - 1,$$

or

$$\frac{(t-1)n+1}{2t+2} > \frac{t}{2} - 1.$$

This is equivalent to

$$t^2 + t - 1 > t^2 - t - 2,$$

which is true for $t \geq 0$. This completes the proof of the lemma. □

Rerouting the edges. Now we shall use the matching provided by Claim 7 to conclude the inductive step. Denote by G' the graph obtained after the completion of T' to a graph of total deficiency $(\text{df}(T') \bmod 2)$. That is, $V(G') = V(T')$ and $E(G') = E(T') \sqcup E(A')$. If $\text{df}(T')$ is odd, G' has a single vertex with deficiency one, denote it by u. For every other vertex $s \in V(G')$, $\text{df}_{G'}(s) = 0$. Our goal is to transform G' into a graph G that will correspond to the graph obtained after the completion of T using only good edge additions.

We initialize G with G'. Let us remind that v is a leaf of T and $T' = T - v$. We denote the only neighbor of v in T by p. Since G is missing vertex v, we introduce v to G, which is now isolated in G. Now $V(G) = V(T)$, so it is left to add missing edges to G, while possibly removing some of the existing edges. Of course, these added edges should include the edge pv, since $E(T) \subseteq E(G)$ must hold. Similarly, we should not remove any edges of T' from G. Thus, we can remove edges in $E(A')$ only. We denote by A the graph of added edges in G, analogously to A' in G'.

As was explained before, our basic technique is to remove the edges of the matching in A', and connect their endpoints to v. However, there are several issues to deal with. First, if p is in $V(A')$, we have to ensure that one of the edges in $E(A')$ incident to p gets removed, otherwise one of the edge additions is wasted on p. This edge removal may in turn disconnect A'. Second, depending on the parity of $\text{df}(T')$ we may have to deal with the already-deficient vertex u of G', and the parity of k comes into play as well. Thus, in the rest of the proof we go over five different cases and show that in each of them

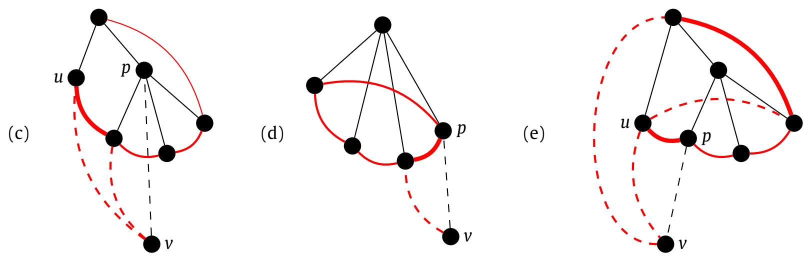

Fig. 1. The cases of rerouting for $k = 3$. Solid edges denote the edges of G'. Straight black edges denote the edges of T', and curved red edges denote the edges of A'. Edges of the matching in A' that are deleted in G are highlighted bold. Dashed edges denote the newly-added edges in G. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

the rerouting is possible. We start with the cases where k is even. Recall that we always start with G being a copy of G' with v as an isolated vertex.

Case (a). *k is even and $p \in V(A')$.* By Claim 7, there is a matching of size $\frac{k}{2}$ in A'. Denote this matching by M. Since $p \in V(A')$, we may assume that $p \in M$ by the following. Suppose that $p \notin M$, p is incident to at least one edge in $E(A')$, say pq. If $q \in M$, remove the edge covering q from M. If $q \notin M$, remove an arbitrary edge from M. Finally, add the edge pq to M. Now M is a matching of size $\frac{k}{2}$ and $p \in M$. We want to remove all edges of M from G, and add all $2 \cdot \frac{k}{2} = k$ edges connecting v with the vertices covered by M. Then the degree of v in G is exactly k, so $\mathrm{df}_G(v) = 0$, no deficiency of any other vertex has changed, and G now contains the edge pv. However, the graph formed by the good edges A must be connected, and this may not be the case since we replace the edge pq by edges pv and qv, and pv belongs to T so p and q are not necessarily connected through the edges of A.

So before replacing the edges of M, we tweak it in order to preserve the connectivity. If p has degree one in A', then after replacing the edges, p is not covered by the good edges anymore. If p has a neighbor t in A' which has degree one in A', then we take the edge pt instead of pq in M. Since v is going to have at least three incident edges, there will be at least one edge of A among them which is not vt, and thus t will be connected to the rest of A. If removing pq makes A disconnected and none of the above is the case, the connected component of p in $A' - pq$ has an edge which is not incident to p. If one of these edges is already in M, then the connectivity will hold since q will be adjacent to v and a vertex in the same component as p will be adjacent to v. Otherwise we remove from M any edge which is not pq and add any edge which belongs to the connected component of p in $A' - pq$ and is not incident to p.

Finally, after rerouting the edges of M the obtained graph G corresponds to an appropriate completion of T. Note that if G' contains the vertex u with deficiency one, it remains deficient in G as well.

Case (b). *k is even and $p \notin V(A')$.* The difference with the previous case is that now we cannot simply ensure that edge pv gets added to G. Add edge pv to G. This does not change the deficiency of p, since it is a vertex of zero deficiency, but deficiency of v now equals $k-1$. Take a matching M of size $\frac{k}{2}$ in A', according to Claim 7. If G contains the unique vertex u of deficiency one, and $u \in M$, remove the edge covering u from M. Otherwise, remove an arbitrary edge from M. M is now a matching of size $\frac{k}{2} - 1$ with $u \notin M$, if u exists. Note that $p \notin M$ necessarily, as $p \notin V(A')$. Remove all edges of M from G, and add all $k - 2$ edges connecting v with vertices covered by M. v is now a vertex of deficiency one in G. If u exists, connect u and v by a new edge. Clearly, this is a good edge addition. This edge did not exist before, because we made sure that $u \notin M$. If u does not exist, v is a single vertex of deficiency one in G. In any case, G is an appropriate completion of T.

We have hereby proved Theorem 4 for the case of even k. We now turn to the cases where k is odd. Clarifying pictures for all three cases to consider are presented in Fig. 1. We start with the easier case.

Case (c). *k is odd and $p \notin V(A')$.* As in Case (b), introduce the edge pv to G, so $\mathrm{df}_G(v) = k-1$, and the deficiency of p does not change. By Claim 7, there is a matching M of size $\frac{k-1}{2}$ in A'. As before, remove all edges of M from G, and add all $k-1$ edges connecting v with the endpoints of edges in M. The vertex v has zero deficiency now, and for any other vertex the deficiency is the same. Clearly, G is an appropriate graph.

Case (d). *k is odd and $p \in V(A')$, there is no deficient vertex in G'.* Take a matching M of size $\frac{k-1}{2}$ in A', and ensure that $p \in M$. In the case $k \geq 5$, do the same tweaking to M as in Case (a) to ensure connectivity. Remove all edges of M from G and connect v to all $k-1$ vertices covered by M. The edge pv is now contained in G, and v is the only vertex with deficiency one in G. This is an appropriate completion of T.

Case (e). *k is odd and $p \in V(A')$, $\mathrm{df}_{G'}(u) = 1$.* Note that $n \geq k+2$ in this case, since for $n = k+1$ there may be no deficient vertices in G'. Take a matching M of size $\frac{k+1}{2}$ in A' according to Claim 7, and ensure that $p \in M$. We consider two cases, either $up \in M$ or not. In the case $up \in M$, if there is another neighbor q of p in A' which is not covered by M, replace up in M by qp, and thus we reduce to the second case. Otherwise, remove all edges of M from G. Now $\deg_G(u) = k-2$, so at least two vertices covered by M are not adjacent to u. At least one of these vertices is distinct from p, denote this vertex by w. Add edges connecting v to each vertex covered by M, except for the vertex w. Thus, v gets connected to exactly $\frac{k+1}{2} \cdot 2 - 1 = k$ vertices, including p and u, so $\mathrm{df}_G(v) = 0$. Finally, add an edge connecting u and w. Now each vertex in G is of zero deficiency. Clearly, each edge addition in this construction is good. To see that A is connected in the case $k \geq 5$, first note that only removing edges up and wt could influence connectivity, where t is such that $wt \in M$. For all other edges of M which were removed, their endpoints are adjacent to v in A, and thus connected in A. The pair w and t remains connected in A since t is adjacent to v, and w is connected to v through u. The vertex p either had the only neighbor u in A', then p is not in $V(A)$, or there was another neighbor q of p in A'. Note that q is necessarily covered by M since we could not replace the edge up. Then q is adjacent to v in A, and p and u are connected through q and v. Also $pv \in E(G)$, so G is an appropriate completion of T.

If $up \notin M$, first remove from M the edge covering u, or if there is no such edge, remove an arbitrary edge from M so that p remains covered by M. Remove all edges of M from G and connect v to all endpoints of these edges as before. Now $\mathrm{df}_G(u) = \mathrm{df}_G(v) = 1$, and $uv \notin G$, so add a good edge between u and v. After that, G is an appropriate completion of T with no deficient vertices. To see that A is connected, we only need to check that p and its pair q in M remain connected. Either u is in the connected component of p in A, and then p and q are connected through u and v, or we do the same tweaking to M as in Case (a) before actually removing the edges.

We considered each case required to accomplish the inductive step. This concludes the proof of Theorem 4. □

Since the class of forests is exactly the class of 1-degenerate graphs, it is reasonable to ask whether Edge k-Core is polynomially solvable on other classes of graphs of bounded degeneracy. The answer is negative, and it was shown by Chitnis and Talmon in [12], where they provided a reduction from Clique to Edge k-Core. We note that they used this reduction to prove that Edge k-Core is W[1]-hard when parameterized by the combined parameter $b + p$, even when $k = 3$.

Proposition 8 *([12]).* Edge k-Core *is* NP*-hard even on the class of* 2*-degenerate graphs for* $k = 3$.

4. Handling small vertex cover

This section is dedicated to Edge k-Core parameterized by the minimum size of a vertex cover of the input graph G. We show that this problem admits an FPT algorithm and complement this result by ruling out the existence of a polynomial kernel. We start with the high level description of the main ideas behind our algorithm.

High-level description of the algorithm. In order to prove that Edge k-Core is FPT parameterized by the vertex cover number of the input graph, we construct an FPT-time Turing reduction from Edge k-Core to instances of integer linear program (ILP), where the number of variables is bounded by some function of the vertex cover. While reducing to ILP is a common approach in the design of parameterized algorithms, see [19, Chapter 6], the reduction for Edge k-Core is not straightforward. In order to make the whole approach applicable, we need a new combinatorial result, Lemma 13. The proof of this lemma strongly exploits the refinement of Tripathi and Vijay [32] of the classical theorem of Erdős and Gallai about degree sequences [20].

The reduction target is the following Integer Linear Programming Feasibility (ILP) problem.

ILP parameterized by ℓ

Input:	Matrix $A \in \mathbb{Z}^{m\times\ell}$ and vector $b \in \mathbb{Z}^m$.
Task:	Is there a vector $x \in \mathbb{Z}^\ell$ such that $A \cdot x \le b$?

ILP is FPT by the celebrated result of Lenstra [16].

Proposition 9 *([17,16,18]).* ILP *can be solved using $\mathcal{O}(\ell^{2.5\ell+o(\ell)} \cdot L)$ arithmetic operations and space polynomial in L. Here L is the number of bits in the input.*

Let G be a simple undirected graph on n vertices and b, k, and p be integers. Let vc be the minimum size of a vertex cover in an n-vertex graph G. Our FPT Turing reduction constructs in time $2^{\mathcal{O}(\mathrm{vc}^2)} \cdot n^{\mathcal{O}(1)}$ at most $2^{\mathcal{O}(\mathrm{vc}^2)}$ instances of ILP. Each instance of ILP has $\ell = 2^{\mathcal{O}(\mathrm{vc})}$ variables. Moreover, at least one of the constructed instances of ILP is a yes-instance if and only if one can build a k-core of size p in G by adding at most b edges. Thus by applying Proposition 9 to each of the instances of ILP, we obtain an FPT (parameterized by vc) algorithm for Edge k-Core.

Recall that in Edge k-Core we are looking for a vertex subset $H \subseteq V(G)$ of size at least p such that $G[H]$ can be completed to a graph of minimum degree at least k using at most b edge additions. In what follows, we describe the reduction from Edge k-Core to ILP.

We start with computing a minimum vertex cover C of G. It is well-known that a simple branching algorithm does this job in time $2^{|C|} \cdot n^{\mathcal{O}(1)}$, see e.g. [19]. We simplify our task a bit by assuming that $C \subseteq H$: we just branch into $2^{|C|}$ possible options of $H \cap C$. For each option we delete vertices $C \setminus H$ from G. We use the following notion of vertex types.

Definition 10 *(Vertex types).* Let G be a graph and C be its vertex cover. For $S \subseteq C$ and a vertex $v \notin C$, we say that *v has type S* if $N_G(v) = S$.

We encode the choice of H (up to isomorphism of $G[H]$) using only $2^{|C|}$ positive integers: for each $S \subseteq C$ we just need to indicate how many vertices of type S are in H. That is, the values of $2^{|C|}$ variables $x_S := |\{v \in H \mid N_G(v) = S, v \notin C\}|$ uniquely define the graph $G[H]$. Then inequality $|C| + \sum_{S\subseteq C} x_S \geq p$ ensures that $|H| \geq p$.

The non-trivial part of the proof is to encode in ILP that $G[H]$ can be completed into a k-core graph using at most b edges. In graph $G[H]$, the vertex set C is a vertex cover and the set $I = H \setminus C$ is an independent set. Assume that $G[H]$ can be completed into a k-core graph by making use of a set of edges B, $|B| \leq b$. The set B can be partitioned into $B = B_C \cup B_I$. Here B_C are the edges with at least one endpoint in C, and $B_I \subseteq \binom{I}{2}$ are the remaining edges. Every edge of B_I has two endpoints in I. We encode the sets B_C and B_I in ILP in different ways.

It is convenient to assume that B_C contains no edges with both endpoints in C. We reach this condition by branching into $2^{\binom{|C|}{2}} = 2^{\mathcal{O}(\mathrm{vc}^2)}$ possible options of which edges between vertices in C are added to G. For each such guess we also update the value b and the conditions on degrees of vertices in C.

The next step in the reduction to ILP is to encode the graph $G[H] + B_C$. Since we do not have edges with both endpoints in C anymore, B_C consists only of edges between C and I. Since C is also a vertex cover of $G[H] + B_C$, there are at most $2^{|C|}$ different types of vertices in $H \setminus C$ in the graph $G[H] + B_C$. A vertex v of type S' in $G[H] + B_C$ has type $S \subseteq S'$ in the graph $G[H]$. Let $y_{S,S'}$ (for $S \subseteq S' \subseteq C$) denote the number of vertices of type S in $G[H]$ that become vertices of type S' in $G[H] + B_C$. Then the set of equations $\sum_{S' \supseteq S} y_{S,S'} = x_S$, for each $S \subseteq C$, ensures that these values correspond to the actual structure of $G[H]$. The cardinality of B_C is then encoded as $\sum_{S'\subseteq C} \sum_{S \subseteq S'} |S' \setminus S| \cdot y_{S,S'}$. Since for each vertex $v \in C$ the graph $G[H] + B_C$ contains all edges incident to v in $G[H] + B$, the resulting degree of v can be checked immediately. Formally, $\deg_{G[C]}(v) + \sum_{S' \ni v} \sum_{S \subseteq S'} y_{S,S'} \geq k$ is equivalent to $\deg_{G[H]+B}(v) \geq k$.

We proceed with the description of how we encode the edge set B_I. For that we need to ensure that for each vertex of I its degree in $G[H] + (B_C \cup B_I)$ is at least k. Since adding edges between vertices in I could significantly increase the vertex cover of $G[H]$, we cannot do the encoding in the same way as for the edges in B_C. However, I remains to be an independent set in $G[H] + B_C$. Therefore, B_I can be any set of edges subject to the condition that in $G[I] + B_I$ the degree of every vertex $v \in I$ is at least $\mathrm{df}_{G[H]+B_C}(v)$. Thus, to ensure that B_I is an appropriate set all we need to consider are the deficiencies of vertices in I.

The deficiencies of vertices in I are integers within the range $[\max\{0, k - |C|\}, k]$. Since $G[I]$ is an empty graph, it is not necessary to know the deficiency of each particular vertex in I. Knowing the number of vertices in I of each particular deficiency is sufficient for our purposes. For $i \in [\max\{0, k - |C|\}, k]$, let s_i denote the number of vertices in I with deficiency i. These variables can be encoded with ILP equations using the variables $y_{S,S'}$.

We arrive at the most interesting and non-trivial part of the reduction. While the inequalities we have built so far are necessary for encoding the information about the set B_I, they are not sufficient. The reason is that not every sequence of integers corresponds to a sequence of vertex degrees in a graph. There is a classical theorem of Erdős and Gallai providing a characterization of graphic sequences. However, if we use this theorem to encode graphic sequences in ILP, the resulting integer program could have unbounded (by a function of vc) number of variables. To overcome this obstacle, we need Lemma 13, a new combinatorial result about graphic sequences.

We want to encode the property that there exists a set of edges B_I of size at most $b - |B_C|$ such that the edges of B_I form a graph with at least s_k vertices of degree at least k, at least s_{k-1} *other* vertices of degree at least $k - 1$, and so on down to $s_{\max\{0,k-|C|\}}$. One technical obstacle here is that we ask for s_i vertices of degree *at least* i, not of degree *exactly* i. In what follows, for clarity, we explain only how to encode the existence of an edge set forming a graph with t_i vertices of degree exactly i for each $i \in [\max\{0, k - |C|\}, k]$. For the "at least" case we need to do more work, but the main idea remains the same. Note that the case we explain here (requiring t_i vertices of degree exactly i) is achieved automatically if all edges in B_I are good edges (that is, consecutive addition of edges from B_I decreases deficiencies of exactly two vertices by one) and the cardinality of this set is found easily as $\frac{1}{2}\sum_i t_i$.

Let us remind the following classical graph-theoretical notion.

Definition 11 *(Graphic sequences).* A sequence $d_1, d_2, \ldots, d_n$ of non-negative integers is called *graphic* if there exists a graph G with $V(G) = \{v_1, v_2, \ldots, v_n\}$, such that $\deg_G(v_i) = d_i$ for each $i \in [n]$.

In terms of this notion, our task is to check that a sequence consisting of integers from $[\max\{0, k - |C|\}, k]$, where the integer i appears exactly t_i times, is a graphic sequence. The problem of determining that a given sequence is graphic was approached by Erdős and Gallai in their famous work [20].

Proposition 12 *(Erdős-Gallai theorem, [20]). A sequence of non-negative integers $d_1 \geq d_2 \geq \ldots \geq d_n$ is graphic if and only if $\sum_{i=1}^n d_i$ is even and $\sum_{i=1}^t d_i \leq t \cdot (t-1) + \sum_{j=t+1}^n \min\{d_j, t\}$ for each $t \in [n]$.*

However, the statement of Proposition 12 does not allow us to encode corresponding inequalities in ILP with the number of variables bounded by $|C|$. We need a refined version of this proposition, Lemma 13. This combinatorial result on graphic sequences of integers in a short range is crucial in constructing ILP inequalities with bounded number of variables. The proof of the lemma is based on the modification of the Erdős-Gallai theorem due to Tripathi and Vijay [32].

Lemma 13. *Let $d_1 \geq d_2 \geq \ldots \geq d_n$ be a sequence of non-negative integers, such that for each $j \in [n]$ $d_j \in [k-a, k]$, for some integers $0 \leq a \leq k < n$. For each $i \in [k-a, k]$, let $t_i = |\{j \mid d_j = i\}|$ be the number of integers equal to i in the sequence. For each $D \in [k-a, k]$, let $T_D = \sum\limits_{i=D}^{k} t_i$.*

Then $d_1, d_2, \ldots, d_n$ is graphic if and only if $\sum_{i=k-a}^{k} i \cdot t_i$ is even and for each $D \in [k-a, k]$ at least one of the following holds:

1. *$T_D < k - a$, or*
2. *$T_D > k$, or*
3. *$\sum\limits_{i=D}^{k} i \cdot t_i \leq T_D \cdot (T_D - 1) + \sum\limits_{i=k-a}^{D-1} \min\{i, T_D\} \cdot t_i$.*

Proof. Clearly, by Proposition 12, the condition that $\sum_{i=k-a}^{k} i \cdot t_i = \sum_{j=1}^{n} d_j$ is even is necessary for any graphic sequence $d_1, \ldots, d_n$.

We now prove that if $d_1, \ldots, d_n$ is graphic, then the third condition (hence, at least one condition) is satisfied for each $D \in [k-a, a]$. Note that the third condition corresponds exactly to the inequality of Erdős and Gallai from Proposition 12 with $t = T_D$, just expressed in the terms of t_i instead of d_j. Hence, if $d_1, d_2, \ldots, d_n$ is graphic, then the third condition is satisfied for each D such that $T_D \in [n]$. The only case left is $T_D = 0$. In this case, the left part of the inequality is equal to zero, so the third condition is also satisfied. It is left to prove the theorem statement in the other direction.

By the result of Tripathi and Vijay [32], to check that $d_1, \ldots, d_n$ is graphic it is sufficient to check the condition of Proposition 12 for all values of t such that $d_t > d_{t+1}$, and for $t = n$. Note that all such values of t are values of T_D for some choices of D. That is, if $d_t > d_{t+1}$, let $D = d_t$. Then $T_D = \sum_{i=D}^{k} t_i = |\{i : d_i \geq D\}| = t$. If we choose $D = k - a$, we achieve $T_D = n$. Thus, to check that $d_1, \ldots, d_n$ is graphic it is enough to check the condition of Proposition 12 for $t = T_D$, for each $D \in [k-a, k]$. This is exactly what the third condition checks, except for the choices of D when one of the first two conditions is satisfied. Hence, it is now left to show that when $T_D < k - a$ or $T_D > k$, then the third condition is satisfied automatically.

First, consider the case $T_D > k$. Note that

$$\sum_{i=D}^{k} i \cdot t_i \leq \sum_{i=D}^{k} k \cdot t_i = k \cdot \sum_{i=D}^{k} t_i = k \cdot T_D \leq (T_D - 1) \cdot T_D.$$

Thus, the third condition is satisfied for $T_D > k$.

Now consider the case $T_D < k - a$. Then

$$\sum_{i=k-a}^{D-1} \min\{i, T_D\} \cdot t_i = \sum_{i=k-a}^{D-1} T_D \cdot t_i = T_D \cdot \sum_{i=k-a}^{D-1} t_i = T_D \cdot (n - T_D).$$

As $k < n$, we have that

$$\sum_{i=D}^{k} i \cdot t_i \leq (n-1) \cdot \sum_{i=D}^{k} t_i = (n-1) \cdot T_D = T_D \cdot (T_D - 1) + T_D \cdot (n - T_D),$$

which is equivalent to the inequality of the third condition of the lemma. Thus, for $T_D < k - a$ the third condition is also satisfied. This concludes the proof of the lemma. □

Lemma 13 still does not yield directly the desired encoding in ILP. Though T_D can be expressed as a sum of t_i's, the summand $T_D \cdot (T_D - 1)$ is not allowed in a *linear* equation with T_D being a variable. However, since the number of T_D's is at most $|C| + 1$, for each T_D the algorithm can guess whether $T_D > k$, $T_D < k - |C|$ or the exact value of $T_D \in [k - |C|, k]$. For each T_D it leads to at most $|C| + 3$ options, so there are at most $|C|^{\mathcal{O}(|C|)}$ possible options in total. This allows us to use the values of T_D's in ILP as constants. Since the variables of type t_i are the only remaining variables, we can write the corresponding constraints as linear inequalities.

We are now ready to state the main result of this section and prove it formally, accumulating the ideas discussed above in this section. The proof also contains the full description of the constructed linear program.

Theorem 14. Edge k-Core *admits an* FPT *algorithm when parameterized by the vertex cover number. The running time of this algorithm is $2^{\mathcal{O}(\mathrm{vc} \cdot 3^{\mathrm{vc}})} \cdot n^{\mathcal{O}(1)}$, where* vc *is the minimum size of a vertex cover of the input n-vertex graph.*

Proof. Let (G, b, k, p) be an instance of Edge k-Core, and let $C \subseteq V(G)$ be a minimum vertex cover in G. Denote $\mathrm{vc} = |C|$. Note that if C is not given as input, it can be found in $2^{\mathrm{vc}} \cdot n^{\mathcal{O}(1)}$ running time with the well-known FPT branching algorithm for Vertex Cover.

EDGE-k-CORE-ILP

achieve

$$(1)\quad \sum_{S'\subseteq C}\sum_{S\subseteq S'}|S'\setminus S|\cdot y_{S,S'}+\sum_{i=\max\{0,k-|C|\}}^{k}\ \sum_{j=\max\{0,k-|C|-1\}}^{i}(i-j)\cdot u_{i,j}+z\le b$$

subject to $x_S\ge 0$, $y_{S,S'}\ge 0$, $s_i\ge 0$, $u_{i,j}\ge 0$, $t_j\ge 0$, $z\ge 0$, and

$$(2)\quad x_S\le |\{v\mid N_G(v)=S,\ v\notin C\}|,\qquad \forall\ S\subseteq C,$$

$$(3)\quad |C|+\sum_{S\subseteq C}x_S\ge p,$$

$$(4)\quad \sum_{S\subseteq S'\subseteq C}y_{S,S'}=x_S,\qquad \forall\ S\subseteq C,$$

$$(5)\quad \deg_{G[C]}(v)+\sum_{S'\ni v}\sum_{S\subseteq S'}y_{S,S'}\ge k,\qquad \forall\ v\in C,$$

$$(6)\quad s_i=\sum_{S'\subseteq C,k-|S'|=i}\ \sum_{S\subseteq S'}y_{S,S'},\qquad \forall\ i\in[k-|C|,k],\ i\ge 0,$$

$$(7)\quad \sum_{j=\max\{0,k-|C|-1\}}^{i}u_{i,j}=s_i,\qquad \forall\ i\in[k-|C|,k],\ i\ge 0,$$

$$(8)\quad t_j=\sum_{i=j}^{k}u_{i,j},\qquad \forall\ j\in[k-|C|-1,k],\ j\ge 0,$$

$$(9)\quad t_j=0,\qquad \forall\ j\in[D_{\max}+1,k],$$

$$(10)\quad t_{D_{\max}}>0,$$

$$(11)\quad D_{\max}\le\sum_{j=\max\{0,k-|C|-1\}}^{k}t_j-1,$$

$$(12)\quad 2z=\sum_{j=\max\{0,k-|C|-1\}}^{D_{\max}}j\cdot t_j,$$

$$(13)\quad \sum_{j=D}^{D_{\max}}t_j=T_D,\qquad \forall\ D\in[L,R],\ \text{if } L\le R,$$

$$(14)\quad \sum_{j=R+1}^{D_{\max}}t_j<k-|C|-1,\qquad \text{if } D_{\max}>R,$$

$$(15)\quad \sum_{j=L-1}^{D_{\max}}t_j>D_{\max},\qquad \text{if } L>\max\{0,k-|C|-1\},$$

$$(16)\quad \sum_{j=D}^{D_{\max}}j\cdot t_j\le T_D(T_D-1)+\sum_{j=\max\{0,k-|C|-1\}}^{D-1}\min\{j,T_D\}\cdot t_j,\qquad \forall\ D\in[L,R].$$

Fig. 2. The ILP formulation of EDGE k-CORE. $D_{\max}$, L, R and T_D's are parameters guessed by the algorithm. Note that equations in the formulation should be replaced with two symmetrical inequalities in order to obtain a valid linear program.

Let $H\subseteq V(G)$ be a subset of the vertices of G such that H becomes a k-core in G after adding at most b edges to G. We can assume that the set of added edges B is a subset of $\binom{H}{2}$, otherwise B is not minimal. Since we are not interested in the vertices outside of H, let our algorithm iterate over all possible values of $H\cap C$, and delete all vertices in $C\setminus H$ from G for a fixed choice of $H\cap C$. Clearly, there are 2^{vc} choices of $H\cap C$, so this adds a factor of 2^{vc} to the running time of the algorithm. We can now assume that we are looking for a k-core that contains all vertices of the vertex cover of G, i.e., $C\subseteq H$. Note that C is a vertex cover in $G[H]$ and denote by $I=H\setminus C$ the independent set in $G[H]$.

Let the algorithm also guess the edges that are added between the vertices of C. There are at most $2^{\binom{\text{vc}}{2}}$ (at most $2^{\mathcal{O}(\text{vc}^2)}$) possible options of adding at most b of these edges. Our algorithm iterates over all possible options of adding edges between the vertices of C, adds these edges in G and decreases the budget b accordingly for each option. This adds a factor of $2^{\mathcal{O}(\text{vc}^2)}$ to the running time of the algorithm. Now, for a fixed such guess, we consider only adding at most b edges to G such that each edge added has at least one endpoint in I.

We are now ready to describe the ILP formulation of EDGE k-CORE. The whole description of EDGE-k-CORE-ILP is presented in Fig. 2.

The formulation uses values $D_{\max}, L, R, T_D$, that are not variables of the linear programming, but are values that the algorithm also needs to guess. To explain the purpose of these values and EDGE-k-CORE-ILP itself, we start directly with the following claim.

Claim 15. *Let $B\subseteq\binom{V(G)}{2}\setminus E(G)\setminus\binom{C}{2}$ be a subset of at most b edges, and let G' denote the graph $G+B$. Let H be the k-core of G'. If $C\subseteq H$ and $|H|\ge p$, then there exist integers $D_{\max}$, L, R and T_D's (for each $D\in[L,R]$), such that* EDGE-k-CORE-ILP *is feasible for this choice of $D_{\max}, L, R, T_D$. Moreover, all these integers are non-negative integers in $[k-|C|-1,k]$.*

Proof. We show how to assign values to the variables of Edge-k-Core-ILP so that every inequality is satisfied. We also explain the purpose of each variable.

For each $S \subseteq C$, assign $x_S = |\{v \in H \setminus C \mid N_G(v) = S\}|$. Thus, x_S denotes the number of vertices of type S in H. Clearly, inequalities (2) are satisfied by such values of x_S. Then (3) counts the vertices in H and ensures that there are at least p of them. This inequality is also satisfied because $|H| \geq p$.

Consider now how the type of a vertex $v \in H \setminus C$ changes in G'. That is, $N_G(v) = S$, but when edges of B are added to G, v may change its type to $N_{G'}(v) = S'$ for some $S' \supseteq S$. Variables $y_{S,S'}$ correspond to the amount of vertices of type S in $G[H]$ that change their type to S' in $G'[H]$. Thus, $y_{S,S'} = \{v \in H \setminus C \mid N_G(v) = S, N_{G'}(v') = S'\}$. Equalities (4) ensure that these amounts agree with the number of vertices of a certain type in $G[H]$, and are clearly satisfied. Note that this handles all edges in B that have one endpoint in I and the other endpoint in C. Thus, one may now evaluate the degree of each $v \in C$ by counting vertices of certain types in $G'[H]$. Inequalities (5) ensure that the degree of each $v \in C$ is at least k. Each one of these inequalities is satisfied since H is a k-core in G'. We now assume that $B \subseteq \binom{H}{2}$, otherwise all edges with an endpoint outside of B can be removed from B while the k-core H in G' and $G'[H]$ remains the same. Hence, $G'[H]$ is exactly the graph $G[H] + B$.

The rest of inequalities of Edge-k-Core-ILP handle edge additions inside $G[I]$. Divide B into two parts, $B = B_C \cup B_I$, where B_C is the set of all edges in B having at least one endpoint in C, and B_I is the set of edges in B having both endpoints in I. After adding all edges in B_C to $G[H]$ we obtain an intermediate graph $G[H] + B_C$. A vertex $v \in H \cap I$ of type S' in $G[H] + B_C$ has deficiency $\mathrm{df}_{G[H]+B_C}(v) = \max\{0, k - |S'|\}$. Then (6) evaluates the number of vertices with certain deficiencies in $G[H] + B_C$. Clearly, all possible deficiency values lie in $[k - |C|, k]$, so s_i indeed denotes the number of vertices with deficiency i for each non-negative $i \in [k - |C|, k]$.

We now consider $B_I \subseteq \binom{I}{2}$. Fix an arbitrary ordering of edges in B_I and consider how edges of B_I are added to $G[H] + B_C$ in this order. Some of these edge additions are good and some are bad. If there are edge additions that do not change deficiency of any vertex, we can remove such edge from B and $G[H] + B$ still remains a k-core. Thus, B_I can be divided into two parts $B_I = B_I^{(2)} \sqcup B_I^{(1)}$, where $B_I^{(2)}$ corresponds to the set of good edges and $B_I^{(1)}$ corresponds to the set of bad edges. Correspondingly, their additions change the total deficiency of $G[H] + B_C$ by two and by one when some order is fixed. Consider the graph $G[I] + B_I^{(2)}$ formed by the good edges in B. For each $v \in I$, $\deg_{G[I]+B_I^{(2)}}(v) \leq \mathrm{df}_{G[H]+B_C}(v)$, as each edge in $B_I^{(2)}$ is good when added to $G[H] + B_C$. Let $t_j = \{v \in I \mid \deg_{G[I]+B_I^{(2)}}(v) = j\}$ be the number of vertices of degree j in $G[I] + B_I^{(2)}$. Then $u_{i,j}$ denotes the number of vertices in I with deficiency i in $G[H] + B_C$ that have degree j in $G[I] + B_I^{(2)}$, and $j \leq i$. That is, $u_{i,j} = \{v \in I \mid \mathrm{df}_{G[H]+B_C}(v) = i, \deg_{G[I]+B_I^{(2)}}(v) = j\}$. This is described by equalities (7) and (8), that are satisfied by the chosen values of t_j and $u_{i,j}$. We only need to show that j is always an integer in $[k - |C| - 1, k]$. If $k \leq |C|$, it is true since $j \geq 0$, so we consider $k > |C|$.

Note that $u_{i,j}$ actually corresponds to adding bad edges from $B_I^{(1)}$. If a vertex $v \in I$ has deficiency $\mathrm{df}_{G[H]+B_C}(v) = i$, but is incident with only $\deg_{G[I]+B_I^{(2)}}(v) = j$ good edges, then it requires $i - j$ bad edges in $B_I^{(1)}$ incident to it to be added in G. This can be considered as follows. Initially, we are given a sequence of vertex deficiencies in I in $G[H] + B_C$. This sequence is described by the values of s_i: for each i, we have s_i vertices in I with deficiency i, so we have s_i integers equal to i in the sequence. By (3), there are at least $p - |C| \geq k + 1 - |C|$ vertices in this sequence. If this sequence is graphic, then we do not require any bad edge addition inside $G[H] + B_C$: just pick good edges as the edges of a graph on $|I|$ vertices corresponding to this sequence. Thus, if the values of s_i correspond to a graphic sequence, $B_I^{(1)}$ can be chosen to be empty. Otherwise, we need to slightly change the sequence of deficiencies using bad edge additions, and we want to minimize these bad edge additions. $u_{i,j}$ specifies how many integers with value i in the sequence are changed to j. This corresponds to adding $i - j$ bad edges incident to each one of $u_{i,j}$ vertices with deficiency i.

Consider the sequence of integers from $[k - |C|, k]$ that is not graphic. With a single operation, we are allowed to decrease some integers of this sequence by one. By Proposition 12, we should always decrease the maximum integer in the sequence by one with a single operation. Suppose that we repeated this operation and obtained integer $k - |C| - 2$ in the sequence at some moment. Then at some moment all integers in the sequence were equal to $k - |C|$, and at some moment later all integers in the sequence were equal to $k - |C| - 1$. Since at least one of $k - |C|$ and $k - |C| - 1$ is even and also there are at least $k + 1 - |C|$ integers in the sequence, there exists a $(k - |C|)$-regular or a $(k - |C| - 1)$-regular graph consisting of $n \geq k + 1 - |C|$ vertices. Thus, at least one of the sequences consisting of integers all-equal to $k - |C|$ or to $k - |C| - 1$ is graphic. Hence, we do not need to decrease any integer below $k - |C| - 1$ in order to minimize the number of operations of changing the sequence.

We have shown that $j \in [k - |C| - 1, k]$. What remains is to explain the purpose of the remaining inequalities of Edge-k-Core-ILP and to show how to choose the values of $D_{\max}, L, R$ and T_D's for each $D \in [L, R]$ in order to satisfy them. The actual purpose of inequalities (9)–(16) is to check that t_j corresponds to a graphic sequence according to Lemma 13. The value of a in Lemma 13 is chosen as $a = \min\{k, |C| + 1\}$. Pick $D_{\max} = \max\{j : t_j > 0\}$. Then (9) and (10) ensure that $D_{\max}$ is chosen correctly. Inequality (11) guarantees that there are at most $D_{\max} + 1$ integers in the sequence, since otherwise it is not graphic. Equality (12) checks that the sum of integers in the sequence is even. Note that $z = |B_I^{(2)}|$ equals the number of good edges in B.

Then, L and R are the bounds of D for which the third condition of Lemma 13 should be verified. Sequence $T_{\max\{0,k-|C|-1\}}, \dots, T_{k-1}, T_k$ is non-increasing, so L is the smallest integer for which $T_L \le k$. Analogously, R is the largest integer for which $T_R \le k-a$. If $i < L$ or $i > R$, then at least one of the first two conditions of Lemma 13 is satisfied, and we can omit checking the Erdős-Gallai inequality for this T_i. Note that it may be the case that $L > R$, then we do not have to check the third condition of Lemma 13 at all. Otherwise, for each $D \in [L, R]$, T_D is a non-negative integer in $[k-|C|-1, k]$. Then (13)–(15) ensure that L, R and T_D's are chosen correspondingly to the values of t_j and are satisfied for our choices of these parameters. Finally, (16) is responsible that the third condition of Lemma 13 holds for each $D \in [L, R]$. Thus, if t_j satisfies (9)–(16) for the correct choice of $D_{\max}, L, R$ and T_D's, then by Lemma 13, t_j corresponds to a graphic sequence. Note that for each choice of t_j there is exactly one feasible choice of the parameters $T_D, D_{\max}, L$, and R.

It is left to explain the role of (1) of EDGE-k-CORE-ILP. We show that the left part of this inequality equals $|B|$. Note that $|B_C| = \sum_{S' \subseteq C} \sum_{S \subseteq S'} |S' \setminus S| \cdot y_{S,S'}$, as edges in B_C can be counted using the number of type changes from S to S', and a type change from S to S' is done with $|S' \setminus S|$ edge additions. The correspondence between bad edges in B_I and the values $u_{i,j}$ is described above, so the second summand in inequality (1) corresponds to $|B_I^{(1)}|$. Finally, the summand z equals $|E(G_I)| = |B_I^{(2)}|$. Thus, the left part of the inequality equals $|B_C| + |B_I^{(1)}| + |B_I^{(2)}| = |B|$, and $|B| \le b$. The proof of the claim is completed. □

We have shown that if G can be completed to a graph with a k-core of size at least p, then EDGE-k-CORE-ILP is feasible for the correct choice of the parameters. Note that each of $D_{\max}, L, R$ and T_D's are integers in $[k-|C|-1, k]$, so there are $|C|+2$ options for each of them. Since there are at most $R-L+1 \le |C|+2$ choices of index D for T_D, there are at most $|C|+5$ integers to choose from $[k-|C|-1, k]$. Clearly, this leads to a total of $\mathcal{O}(|C|^{|C|})$ possible options for the parameters. The algorithm iterates over all possible choices of these values, constructs an ILP formulation for each fixed choice and then employs the algorithm of Proposition 9 to check if EDGE-k-CORE-ILP is feasible. There are $\mathcal{O}(3^{|C|})$ variables and inequalities in the formulation (with $y_{S,S'}$ being the majority of the variables), so the feasibility of the formulation is checked in $\mathcal{O}(3^{|C|})^{\mathcal{O}(3^{|C|})} \cdot n^{\mathcal{O}(1)} = 3^{\mathcal{O}(|C| \cdot 3^{|C|})} \cdot n^{\mathcal{O}(1)}$. The outer guesses of $H \cap C$, $B \cap \binom{C}{2}$ and the parameters of ILP add a factor of $|C|^{|C|} \cdot n^{\mathcal{O}(1)}$ to the running time of the algorithm. This factor is dominated by the double exponent, so the total running time is still $3^{\mathcal{O}(|C| \cdot 3^{|C|})} \cdot n^{\mathcal{O}(1)}$. It is only left to say that if the algorithm finds EDGE-k-CORE-ILP feasible for some choice of parameters, then the initial instance is a yes-instance of EDGE k-CORE.

Claim 16. *If* EDGE-k-CORE-ILP *is feasible for some choice of* $D_{\max}, L, R, T_D$, *then* (G, b, k, p) *is a yes-instance of* EDGE k-CORE.

Proof. One needs to follow the proof of the previous claim. It can be shown that from a feasible solution to EDGE-k-CORE-ILP can be constructed a set of vertices $H \subseteq V(G)$ (following the values of x_S) and sets of edges B_C (following the values of $y_{S,S'}$), $B_I^{(1)}$ (following the values of $u_{i,j}$) and $B_I^{(2)}$ (following that the values of t_j correspond to a graphic sequence). Then $G[H]$ can be completed to a graph of minimum degree k by adding edges from $B = B_C \cup B_I^{(1)} \cup B_I^{(2)}$ to G. Inequalities ensure that $|H| \ge p$ and $|B| \le b$. Note that the constructed H is not necessarily the k-core of G', but it is necessarily a subset of its vertices. □

This completes the proof of Theorem 14. □

To complement our FPT algorithm, we show that EDGE k-CORE does not admit a polynomial kernel when parameterized by the combined parameter $\mathrm{vc} + k + b + p$. It was shown in [33] that the BOUNDED RANK DISJOINT SETS problem does not admit a polynomial kernel, and our proof is by reduction from this problem.

BOUNDED RANK DISJOINT SETS

Input:	A family $\mathcal{F}$ over a universe U with every set $S \in \mathcal{F}$ having size at most d together with a positive integer k.
Task:	Does there exist a subfamily $\mathcal{F}'$ of $\mathcal{F}$ with $|\mathcal{F}'| \ge k$ such that for every pair of sets $S_1, S_2 \in \mathcal{F}'$ we have $S_1 \cap S_2 = \emptyset$?

Proposition 17 *([33]).* BOUNDED RANK DISJOINT SETS *does not admit a polynomial kernel when parameterized by $d + k$ even restricted to instances where $|U| = dk$ and for every $S \in \mathcal{F}$ the size of S is exactly d, unless* NP $\subseteq$ coNP/poly.

Note that the result above is originally stated under the assumption that the polynomial hierarchy does not collapse to the third level ($\mathsf{PH} \ne \Sigma_p^3$) [33]. Here we state the results under the weaker assumption that NP $\not\subseteq$ coNP/poly, as it is common in the modern literature on kernelization [21]. Proposition 17 in the stated form follows from the following: Dom et al. [33] show that the problem admits an OR-composition, which immediately implies that the problem admits a cross-composition into itself. By employing the general result on cross-compositions [21, Theorem 17.8], this fact gives

Proposition 17. For more details on the framework for showing infeasibility of polynomial kernels see Bodlaender et al. [34], Fortnow and Santhanam [35], and the recent book on kernelization [21].

Now we show our reduction.

Theorem 18. *Unless* NP $\subseteq$ coNP/ poly, Edge k-Core *does not admit a polynomial kernel when parameterized by the combined parameter* vc $+ k + b + p$.

Proof. We provide a polynomial reduction from Bounded Rank Disjoint Sets. Consider an instance $(U, \mathcal{F}, k, d)$ of Bounded Rank Disjoint Sets such that $|U| = dk$ and all the sets in $\mathcal{F}$ have size exactly d. We may assume that k is even, otherwise extend the universe by d new elements, add one set containing all of these elements to $\mathcal{F}$, and increase k by one. Clearly, the new set must always be in the resulting subfamily, and thus the transformed instance has a solution if and only if the original instance has a solution.

Now we construct an instance (G, b, k', p) of Edge k-Core. The vertices of G consist of set vertices, one for each set in $\mathcal{F}$, universe vertices, one for each element of the universe, and d clique vertices. The clique vertices are pairwise connected, each of the universe vertices is connected to each of the clique vertices, and for each set $S \in \mathcal{F}$ the vertex corresponding to S is connected to the universe vertices corresponding to the elements contained in S. Formally,

$$V(G) = V_{\mathcal{F}} \cup V_U \cup V_C = \{v_S : S \in \mathcal{F}\} \cup \{v_u : u \in U\} \cup \{v_{c_i} : i \in \{1, \cdots, d\}\},$$
$$E(G) = \{v_S v_u : S \in \mathcal{F}, u \in S\} \cup \{v_u v_{c_i} : u \in U, i \in \{1, \cdots, d\}\} \cup \{v_{c_i} v_{c_j} : 1 \le i < j \le d\}.$$

Finally, we set $b = k/2$, $k' = d + 1$ and $p = d + |U| + k$. Since $V_U \cup V_C$ is a vertex cover of G, vc $\le |U| + d = kd + d$, so the combined parameter vc $+ k' + b + p$ is indeed bounded by a polynomial in $k + d$.

Next, we show that (G, b, k', p) is a yes-instance of Edge k-Core if and only if $(U, \mathcal{F}, k, d)$ is a yes-instance of Bounded Rank Disjoint Sets. First, assume there is a subfamily $\mathcal{F}' \subset \mathcal{F}$ such that $|\mathcal{F}'| \ge k$ and for each pair of sets S_1, S_2 from $\mathcal{F}'$, $S_1 \cap S_2 = \emptyset$. Since each set in $\mathcal{F}$ contains exactly d elements and $|U| = kd$, $|\mathcal{F}'| = k$ and $\cup_{S \in \mathcal{F}'} S = U$. Consider the following vertex set H in G: $H = V_C \cup V_U \cup \{v_S : S \in \mathcal{F}'\}$, and an arbitrary matching $B \subset \binom{V(G)}{2} \setminus E(G)$ of size b on vertices $\{v_S : S \in \mathcal{F}'\}$. Clearly, B exists since $V_{\mathcal{F}}$ is an independent set in G and $|\mathcal{F}'| = k$ which is even; also, $|B| = k/2 = b$. Let G' denote the graph $G + B$. We claim that H induces a k'-core in G'. Each of the set vertices in H is adjacent to d universe vertices, and also to another set vertex by an edge from B, so its degree in $G'[H]$ is $d + 1$. Each of the universe vertices is adjacent to one set vertex in H, since $\cup_{S \in \mathcal{F}'} S = U$, and also to d clique vertices, so its degree in $G'[H]$ is also $d + 1$. Each of the clique vertices is adjacent to $d - 1$ clique vertices and dk universe vertices, so its degree in $G'[H]$ is $dk + d - 1 \ge d + 1$, since $dk \ge 2$ as k is even. The size of H is $d + |U| + k = p$, so B is a solution to (G, b, k', p).

In the other direction, assume there exist $H \subset V(G)$ and $B \subset \binom{V(G)}{2} \setminus E(G)$ such that $|H| \ge p$, $|B| \le b$ and H induces a k'-core in G', where G' denotes the graph $G + B$. Since set vertices have degree d in G, $|H \cap V_{\mathcal{F}}|$ must be at most $2b = k$ as each vertex in $H \cap V_{\mathcal{F}}$ must have at least one incident edge in B. Since $|H| \ge p = d + |U| + k$, H contains V_C and V_U, and $|H \cap V_{\mathcal{F}}| = k$. Then each of the vertices in $H \cap V_{\mathcal{F}}$ has exactly one incident edge in B, and so B is a matching on $H \cap V_{\mathcal{F}}$. For each $v_u \in V_U$ there exists an adjacent $v_S \in H \cap V_{\mathcal{F}}$, otherwise $\deg_{G'[H]}(v_u) = d < k'$ as v_u is adjacent only to V_C in G, and no edge in B is incident to v_u. Then, $\mathcal{F}' = \{S : v_S \in H \cap V_{\mathcal{F}}\}$ covers the whole universe, and so $\mathcal{F}'$ is a solution to $(U, \mathcal{F}, k, d)$, since $|\mathcal{F}'| = k$, $|U| = kd$ and thus the sets in $\mathcal{F}'$ must be disjoint. This finishes the proof of the theorem. □

5. Exploiting the treewidth

In this section, we give an FPT-algorithm for Edge k-Core parameterized by tw $+ k$. This improves upon the following result of Chitnis and Talmon, and we also use their algorithm as a subroutine.

Proposition 19 *([12]).* Edge k-Core *can be solved in time* $(k + \text{tw})^{\mathcal{O}(\text{tw}+b)} \cdot n^{\mathcal{O}(1)}$.

We start with the central combinatorial result of this section that allows for the algorithmic improvement. Namely, we show that whenever the total deficiency of a graph G exceeds a polynomial in k, G can always be completed to a graph of minimum degree k using the minimum possible number of edges with the given deficiency. Also, the required edge additions can be identified in polynomial time.

We believe that this result is interesting on its own, since it considerably simplifies the problem whenever the budget is sufficiently high compared to k. If we are trying to identify the best vertex set H which induces a k-core, we have to only care about the total deficiency of $G[H]$, and not of any particular structure on it.

Theorem 20. *For any integer $k \ge 2$, any graph G with* df$(G) \ge 3k^3$ *can be completed to a graph of minimum degree k using* $\lceil \frac{1}{2} \text{df}(G) \rceil$ *edge additions in polynomial time.*

Proof. It is enough to prove that we can satisfy all deficiencies by adding only good edges, except if df(G) is odd, exactly one edge addition is bad.

We constructively obtain a graph G' of form $G + B$, where initially $B = \emptyset$. The construction is a polynomial time algorithm.

First, we exhaustively apply the following rule, which always adds one good edge. If there are two distinct vertices $u, v \in V(G')$ such that $\mathrm{df}_{G'}(u) > 0$, $\mathrm{df}_{G'}(v) > 0$, and $uv \notin E(G')$, then add the edge uv to B. Assume that the rule is no longer applicable. Let us denote $C = \{v \in V(G) | \mathrm{df}_{G'}(v) > 0\}$, by the conditions of the rule, C induces a clique in G'. Then, $|C| \le k$, since otherwise vertices in C could not have positive deficiency.

Now we exhaustively apply the following second rule. Fix two vertices $u, v \in C$, such that either u and v are distinct, or $u = v$ and $\mathrm{df}_{G'}(u) \ge 2$. Then find two distinct vertices $u', v' \in V(G') \setminus C$ such that $u'v' \in B$ and $uu', vv' \notin E(G')$. Since $u'v'$ is in B, u' and v' have degree exactly k, as previously we have only added good edges and $u', v' \notin C$. Delete $u'v'$ from B, now u' and v' have positive deficiencies. Add edges uu' and vv' to B, by the choice of u' and v' these edges are not in $E(G')$, and also both these additions are good.

We claim that when the second rule is no longer applicable, the size of C is at most one, and $\mathrm{df}(G')$ is also at most one. Suppose it is not true, in this case there is always a proper choice of $u, v \in C$. Then there are no $u', v' \in V(G) \setminus C$ satisfying the conditions above. Then each edge $u'v' \in B$ is of one of the following kinds:

1. $u', v' \in C$, since $|C| \le k$, there are at most $\binom{k}{2}$ such edges;
2. one of u', v' is in C and the other is not in C, there are at most $k(k-1)$ edges of this kind, since $|C| \le k$ and degrees in C are less than k;
3. $u', v' \notin C$, and either $uu' \in E(G')$ or $vv' \in E(G')$; there are at most $k(k-1)$ vertices adjacent to C, and each of them has at most k incident edges from B, so there are at most $k^2(k-1)$ such edges.

Then the size of B is at most $\binom{k}{2} + k(k-1) + k^2(k-1) < 2k^3$. However, $\mathrm{df}(G) = 2|B| + \mathrm{df}(G')$, and $\mathrm{df}(G') \le |C| \cdot k \le k^2$. So $\mathrm{df}(G) < 3k^3$ contradicting the statement.

Therefore, by the constructed sequence of good additions we reached the situation where $|C|$ and $\mathrm{df}(G')$ are both at most one. If C is empty, we are done. If C consists of one vertex u, then its deficiency is one. Since $\mathrm{df}(G) = 2|B| + \mathrm{df}(G')$, $\mathrm{df}(G)$ is odd, and we have one more edge addition. Then we add to B an edge from u to any other vertex v such that $uv \notin E(G')$; this is always possible since $\deg_{G'}(u) < k$, and $V(G) > k$ because $\mathrm{df}(G) \ge 3k^3$. $\square$

The intuition to our FPT algorithm is as follows. When we can obtain a sufficiently large k-core by adding a number of edges $b < 3k^3$, the algorithm from Proposition 19 suffices. Otherwise $b \ge 3k^3$ and by Theorem 20 we can focus on finding a vertex subset in G of size at least p minimizing the total deficiency of the induced subgraph. We show how to do that with a dynamic programming over a tree decomposition. First we introduce some preliminaries about treewidth and tree decompositions from [19]. A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, where T is a tree whose every node t is assigned a vertex subset $X_t \subseteq V(G)$, called a bag, such that the following three conditions hold:

(T1) $\bigcup_{t \in V(T)} X_t = V(G)$.
(T2) For every $uv \in E(G)$, there exists a node t of T such that bag X_t contains both u and v.
(T3) For every $u \in V(G)$, the set $T_u = \{t \in V(T) : u \in X_t\}$, i.e., the set of nodes whose corresponding bags contain u, induces a connected subtree of T.

The *width* of tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ equals $\max_{t \in V(T)} |X_t| - 1$, that is, the maximum size of a bag in $\mathcal{T}$ minus 1. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum possible width of a tree decomposition of G.

For simplicity, we will only consider so-called *nice tree decompositions*. A tree decomposition $(T, \{X_t\}_{t \in V(T)})$ rooted at $r \in T$ is *nice* if the following conditions are satisfied:

- $X_r = \emptyset$ and $X_\ell = \emptyset$ for every leaf ℓ of T. In other words, all the leaves as well as the root contain empty bags.
- Every non-leaf node of T is of one of the following three types:
  - **Introduce node**: a node t with exactly one child t' such that $X_t = X_{t'} \cup \{v\}$ for some vertex $v \notin X_{t'}$; we say that v is *introduced* at t.
  - **Forget node**: a node t with exactly one child t' such that $X_t = X_{t'} \setminus \{w\}$ for some vertex $w \in X_{t'}$; we say that w is *forgotten* at t.
  - **Join node**: a node t with two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.

As shown in [19], any tree decomposition can be turned into a nice one in polynomial time. For a node t in a nice tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ denote by V_t the union of all the bags present in the subtree of T rooted at t, including X_t.

Proposition 21 *([19]). Given a graph G of treewidth* tw, *a nice tree decomposition of an n-vertex graph G of width* $4\mathrm{tw} + 4$ *can be constructed in time* $8^{\mathrm{tw}} \cdot n^{\mathcal{O}(1)}$.

Now we are ready to show our algorithm.

Lemma 22. *Given an n-vertex graph G of treewidth tw and integers k, p, the value*

$$\min\{\mathrm{df}(G[\widehat{S}]) : \widehat{S} \subseteq V(G), |\widehat{S}| \geq p\}$$

can be computed in time $k^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$.

Proof. Consider a nice tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G given by the algorithm in Proposition 21, width of $\mathcal{T}$ is $\mathcal{O}(\mathrm{tw})$. For every node t, every $S \subset X_t$, every function $f : S \to \{0, \cdots, k\}$, and every number $s \in \{0, \cdots, |V_t|\}$ define the following value:

$$c[t, S, f, s] = \min\{\mathrm{df}(G[\widehat{S}]) : \widehat{S} \in \mathcal{C}[t, S, f, s]\},$$

where

$$\mathcal{C}[t, S, f, s] = \{\widehat{S} \subset V_t : \widehat{S} \cap X_t = S, |\widehat{S}| = s, \text{ and for every } v \in S, \mathrm{df}_{G[\widehat{S}]}(v) = f(v)\}.$$

Next, we show how the values of $c[\cdot, \cdot, \cdot, \cdot]$ are computed for all kinds of nice tree decomposition nodes.

Leaf node. If t is a leaf node, there is only one value $c[t, \emptyset, f_\emptyset, 0] = 0$, where $f_\emptyset$ denotes the empty function to $\{0, \cdots, k\}$.

Introduce node. Let t be an introduce node, t' be its only child and v be the vertex such that $v \notin X_{t'}$, $X_t = X_{t'} \cup \{v\}$. We claim that for any S, f, s from the definition of $c[t, \cdot, \cdot, \cdot]$

$$c[t, S, f, s] = \begin{cases} c(t', S, f, s) & \text{if } v \notin S; \\ c(t', S \setminus \{v\}, f|_{S \setminus \{v\}}, s-1) + f(v) & \text{if } v \in S, f(v) = \mathrm{df}_{G[S]}(v), s > 0; \\ \infty & \text{otherwise}. \end{cases} \tag{9}$$

If $v \notin S$, (9) clearly holds since the families of the sets $\widehat{S}$ are the same in both parts of the equation. If $v \in S$, then for $\widehat{S} \in \mathcal{C}[t, S, f, s]$ to exist s must be greater than zero since $v \in \widehat{S}$. Also, $f(v)$ must be equal to $\mathrm{df}_{G[S]}(v)$ since there are no edges from v to $V_t \setminus X_t$ by the definition of a nice tree decomposition, and thus $\mathrm{df}_{G[S]}(v) = \mathrm{df}_{G[\widehat{S}]}(v)$. This shows that the third case of (9) holds, so only the second case remains.

When the conditions of the second case hold, we show that for each $\widehat{S}$ in $\mathcal{C}[t, S, f, s]$ there is a corresponding $\widehat{S}'$ in $\mathcal{C}(t', S \setminus \{v\}, f|_{S \setminus \{v\}}, s-1)$, this is a one-to-one correspondence, and $\mathrm{df}(G[\widehat{S}]) = \mathrm{df}(G[\widehat{S}']) + \mathrm{df}_{G[S]}(v)$. Namely, $\widehat{S}' = \widehat{S} \setminus \{v\}$, and this concludes the proof of (9).

Forget node. Let t be a forget node, t' be its only child and v be the vertex such that $v \notin X_t$, $X_{t'} = X_t \cup \{v\}$. We claim that for any S, f, s from the definition of $c[t, \cdot, \cdot, \cdot]$

$$c[t, S, f, s] = \min\{c(t', S, f, s)\} \cup \{c(t', S \cup \{v\}, f', s) : f'|_S = f\}. \tag{10}$$

First, consider any $\widehat{S} \in \mathcal{C}[t, S, f, s]$. If $v \notin \widehat{S}$, then $\widehat{S}$ is also in $\mathcal{C}(t', S, f, s)$. If $v \in \widehat{S}$, then $\widehat{S}$ is in $\mathcal{C}[t', S \cup \{v\}, f', s]$, where $f'|_S = f$ and $f'(v) = \mathrm{df}_{G[\widehat{S}]}(v)$.

In the other direction, if $\widehat{S} \in \mathcal{C}[t', S, f, s]$ or $\widehat{S} \in \mathcal{C}[t', S \cup \{v\}, f', s]$ where $f'|_S = f$, then $\widehat{S} \in \mathcal{C}[t, S, f, s]$.

Join node. Let t be a join node, t_1 and t_2 be its children, such that $X_t = X_{t_1} = X_{t_2}$. We claim that for any S, f, s from the definition of $c[t, \cdot, \cdot, \cdot]$

$$c[t, S, f, s] = \min_{s_1 + s_2 = s + |S|, f_1, f_2} \left(c(t_1, S, f_1, s_1) + c(t_2, S, f_2, s_2) + \sum_{v \in S} (f(v) - f_1(v) - f_2(v)) \right), \tag{11}$$

where f_1 and f_2 are such that for every $v \in S$, $f(v) = \max(0, f_1(v) + f_2(v) - k - \deg_{G[S]}(v))$. To show the correctness of (11), first we show that the right side of (11) is at most the left side. Consider an $\widehat{S} \in \mathcal{C}[t, S, f, s]$ on which the minimum is attained, i.e., $c[t, S, f, s] = \mathrm{df}(G[\widehat{S}])$. Denote $\widehat{S}_1 = \widehat{S} \cap V_{t_1}$, $\widehat{S}_2 = \widehat{S} \cap V_{t_2}$, $s_1 = |\widehat{S}_1|$, $s_2 = |\widehat{S}_2|$. Since $\widehat{S}_1 \cap \widehat{S}_2 = \widehat{S} \cap V_{t_1} \cap V_{t_2} = \widehat{S} \cap X_t = S$, $s_1 + s_2 = s + |S|$ holds. Define also f_1 and f_2 as $f_1(v) = \mathrm{df}_{G[\widehat{S}_1]}(v)$, $f_2(v) = \mathrm{df}_{G[\widehat{S}_2]}(v)$. By definition, $\widehat{S}_1 \in \mathcal{C}[t_1, S, f_1, s_1]$ and $\widehat{S}_2 \in \mathcal{C}[t_2, S, f_2, s_2]$. Since for every $v \in S$

$$\begin{aligned} \mathrm{df}_{G[\widehat{S}]}(v) &= \max(0, k - \deg_{G[\widehat{S}]}(v)) = \\ &\max(0, k - \deg_{G[\widehat{S}_1]}(v) - \deg_{G[\widehat{S}_2]}(v) + \deg_{G[S]}(v)) = \\ &\max(0, \mathrm{df}_{G[\widehat{S}_1]}(v) + \mathrm{df}_{G[\widehat{S}_2]}(v) - k - \deg_{G[S]}(v)) = \\ &\max(0, f_1(v) + f_2(v) - k - \deg_{G[S]}(v)), \end{aligned} \tag{12}$$

and $f(v) = \mathrm{df}_{G[\widehat{S}]}(v)$, the condition on f, f_1 and f_2 holds. Thus, the right side of (11) is at most

$$\mathrm{df}(G[\widehat{S}_1]) + \mathrm{df}(G[\widehat{S}_2]) + \sum_{v\in S}(f(v) - f_1(v) - f_2(v)) =$$
$$\mathrm{df}(G[\widehat{S}_1]) - \sum_{v\in S}\mathrm{df}_{G[\widehat{S}_1]}(v) + \mathrm{df}(G[\widehat{S}_2]) - \sum_{v\in S}\mathrm{df}_{G[\widehat{S}_2]}(v) + \sum_{v\in S}\mathrm{df}_{G[\widehat{S}]}(v) =$$
$$\sum_{v\in \widehat{S}_1\setminus S}\mathrm{df}_{G[\widehat{S}_1]}(v) + \sum_{v\in \widehat{S}_2\setminus S}\mathrm{df}_{G[\widehat{S}_2]}(v) + \sum_{v\in S}\mathrm{df}_{G[\widehat{S}]}(v) =$$
$$\sum_{v\in \widehat{S}}\mathrm{df}_{G[\widehat{S}]}(v) = \mathrm{df}(G[\widehat{S}]), \quad (13)$$

since $(\widehat{S}_1 \setminus S) \uplus (\widehat{S}_2 \setminus S) \uplus S = \widehat{S}$, and for every $v \in \widehat{S}_1 \setminus S$, $\mathrm{df}_{G[\widehat{S}_1]}(v) = \mathrm{df}_{G[\widehat{S}]}(v)$ as v is adjacent only to vertices in $\widehat{S}_1$, the analogous holds for every $v \in \widehat{S}_2 \setminus S$. Since $\mathrm{df}(G[\widehat{S}]) = c[t, S, f, s]$, the first part of the correctness proof is done.

Second, we show that the left side of (11) is at most the right side of (11). Consider s_1, s_2, f_1, f_2 on which the minimum of the left side of (11) is attained, and consider also $\widehat{S}_1 \in \mathcal{C}[t_1, S, f_1, s_1]$ and $\widehat{S}_2 \in \mathcal{C}[t_2, S, f_2, t_2]$ such that $\mathrm{df}(G[\widehat{S}_1]) = c[t_1, S, f_1, s_1]$ and $\mathrm{df}(G[\widehat{S}_2]) = c[t_2, S, f_2, s_2]$. Let $\widehat{S} = \widehat{S}_1 \cup \widehat{S}_2$, we claim that $\widehat{S} \in \mathcal{C}[t, S, f, s]$. Since $X_t = X_{t_1} = X_{t_2}$ and $\widehat{S}_1 \cap X_{t_1} = \widehat{S}_2 \cap X_{t_2} = S$, $\widehat{S} \cap X_t$ is also S. Since $\widehat{S}_1 \cap \widehat{S}_2 = S$, $|\widehat{S}| = |\widehat{S}_1| + |\widehat{S}_2| - |S| = s_1 + s_2 - |S| = s$. The equation (12) holds with the new choice of $\widehat{S}$, $\widehat{S}_1$, and $\widehat{S}_2$; thus, $\mathrm{df}_{G[\widehat{S}]}(v) = f(v)$ for every $v \in S$, and $\widehat{S} \in \mathcal{C}[t, S, f, s]$. Finally, (13) also holds, and $\mathrm{df}(G[\widehat{S}])$ is equal to the value of the right side of (11). Thus the correctness of (11) is fully proven.

We compute all values $c[\cdot,\cdot,\cdot,\cdot]$ by going through nodes of T in the tree order, starting from the leaves, and applying one of the above formulas, depending on the type of node t, for all possible values of S, f, s at this node. Finally, at the root node r for every number $s \in \{0, \cdots, n\}$ we have the value $c[r, \emptyset, f_\emptyset, s]$, which is equal to $\min_{\widehat{S}\subset V(G), |\widehat{S}|=s} \mathrm{df}(G[\widehat{S}])$. Finally, we return the value $\min_{p\le s\le n} c[r, \emptyset, f_\emptyset, s]$.

For each of the $\mathcal{O}(kn)$ nodes of t, there are $2^{|X_t|}$ variants of S, at most $(k+1)^{|X_t|}$ variants of f, and $n+1$ variants of s in the definition of $c[t, S, f, s]$. So the total number of values computed is $k^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$, and the total running time is also $k^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$. □

With all pieces together, we are ready to prove the main algorithmic result of this section.

Theorem 23. Edge k-Core *admits an* FPT *algorithm when parameterized by the combined parameter* tw + k.

Proof. Consider the instance (G, b, k, p) of Edge k-Core. Assume $p > k$ since the size of a k-core is always at least $k + 1$.

If $b \le 3k^3$, we run the algorithm of Chitnis and Talmon given by Proposition 19 and return its result. Otherwise, we run the algorithm from Lemma 22 on the input (G, k, p), and obtain the value d such that there is a $\widehat{S}^* \subset V(G)$ of size at least p with $\mathrm{df}(G[\widehat{S}^*]) = d$, and $\mathrm{df}(G[\widehat{S}]) \ge d$ for every $\widehat{S} \subseteq V(G)$ of size at least p.

Now, if $d < 3k^3$, we report a yes-instance, since $b \ge 3k^3$, and $G[\widehat{S}^*]$ can be trivially turned into a k-core. Namely, for each $v \in \widehat{S}^*$, while $\deg_{G[\widehat{S}^*]} < k$ add an edge connecting v and some other vertex in $\widehat{S}^*$ arbitrarily. This is always possible since $|\widehat{S}^*| > k$, and the number of edges added is at most d.

If $d \ge 3k^3$, we compare b to $\lceil \frac{1}{2}d \rceil$. If $b \ge \lceil \frac{1}{2}d \rceil$, we report a yes-instance, since $G[\widehat{S}^*]$ can be turned into a k-core using $\lceil \frac{1}{2}\mathrm{df}(G[\widehat{S}^*]) \rceil$ edge additions by Lemma 20. If $b < \lceil \frac{1}{2}d \rceil$, we report a no-instance, since for every $\widehat{S} \subset V(G)$ completing $G[\widehat{S}]$ to a k-core requires at least $\lceil \frac{1}{2}\mathrm{df}(G[\widehat{S}]) \rceil$ edge additions, and $d \le \mathrm{df}(G[\widehat{S}])$ for every $\widehat{S}$ of size at least p.

It remains to bound the running time. In the case $b \le 3k^3$, the running time is $(k + \mathrm{tw})^{\mathcal{O}(k^3+\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ by Proposition 19. In the case $b \ge 3k^3$ the algorithm from Lemma 22 dominates the running time, which is $k^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$. Thus, the whole algorithm is FPT parameterized by tw + k. □

6. Conclusion and open questions

In this work, we investigated the complexity of Edge k-Core on several sparse graph classes: forests, graphs with bounded vertex cover and graphs of bounded treewidth. There are, however, several natural questions that remain open. First, while we show an FPT algorithm when parameterized by treewidth plus k, we are not aware of any hardness result excluding an FPT algorithm parameterized by treewidth only. In fact, for any structural parameter "between" vertex cover and treewidth, e.g. feedback vertex set or treedepth, this question remains open. Second, while known NP-hardness reductions for Edge k-Core work even for constant values of k and the degeneracy of the graph, it would be interesting to know whether Edge k-Core admits an FPT algorithm when parameterized by k plus degeneracy plus some other parameter, e.g. the budget b. Third, we are not aware of a lower bound matching the double exponential running time of our vertex cover algorithm. It would be interesting to see whether there is a more efficient FPT algorithm in this parameterization, or a non-trivial fine-grained hardness result. Finally, our work leaves wide open the complexity of Edge k-Core on dense graph classes. Is there an FPT algorithm for Edge k-Core parameterized by, say, cliquewidth or neighborhood diversity, possibly, together with another parameter like k or b? Based on our results, we can only project tractability for very special "dense" parameters. For example, FPT algorithm parameterized by the vertex deletion distance to a bounded number of disjoint cliques is likely to follow along the same lines as our FPT by vertex cover algorithm, with additional variables capturing how many vertices from a particular neighborhood class of a particular clique are taken into the core.

CRediT authorship contribution statement

Fedor V. Fomin: Funding acquisition, Investigation, Project administration, Supervision, Writing – original draft. **Danil Sagunov:** Investigation, Writing – original draft, Writing – review & editing. **Kirill Simonov:** Investigation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] F.V. Fomin, D. Sagunov, K. Simonov, Building large k-cores from sparse graphs, in: J. Esparza, D. Král' (Eds.), 45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, Prague, Czech Republic, August 24–28, 2020, in: LIPIcs, vol. 170, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 35.

[2] K. Bhawalkar, J.M. Kleinberg, K. Lewi, T. Roughgarden, A. Sharma, Preventing unraveling in social networks: the anchored k-core problem, SIAM J. Discrete Math. 29 (3) (2015) 1452–1475, https://doi.org/10.1137/14097032X.

[3] M.S. Chwe, Structure and strategy in collective action, Am. J. Sociol. 105 (1) (1999) 128–156, http://www.jstor.org/stable/10.1086/210269.

[4] M.S.-Y. Chwe, Communication and coordination in social networks, Rev. Econ. Stud. 67 (1) (2000) 1–16, http://www.jstor.org/stable/2567025.

[5] S. Wuchty, E. Almaas, Peeling the yeast protein network, Proteomics 5 (2) (2005) 444–449, https://doi.org/10.1002/pmic.200400962.

[6] J.I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, A. Vespignani, K-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases, Netw. Heterog. Media 3 (2) (2008) 371–393, https://doi.org/10.3934/nhm.2008.3.371.

[7] J.I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, A. Vespignani, Large scale networks fingerprinting and visualization using the k-core decomposition, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), Advances in Neural Information Processing Systems (NIPS), vol. 18, MIT Press, 2005, pp. 41–50.

[8] J.I. Alvarez-Hamelin, M.G. Beiró, J.R. Busch, Understanding edge connectivity in the internet through core decomposition, Internet Math. 7 (1) (2011) 45–66, https://doi.org/10.1080/15427951.2011.560786.

[9] C. Giatsidis, F. Malliaros, D. Thilikos, M. Vazirgiannis, Corecluster: a degeneracy based graph clustering framework, Proc. AAAI Conf. Artif. Intell. 28 (1) (2014), https://doi.org/10.1609/aaai.v28i1.8731, https://ojs.aaai.org/index.php/AAAI/article/view/8731.

[10] M. Burke, C. Marlow, T.M. Lento, Feed me: motivating newcomer contribution in social network sites, in: Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI), ACM, 2009, pp. 945–954.

[11] T. Schelling, Micromotives and Macrobehavior, Fels Lectures on Public Policy Analysis, W. W. Norton & Company, 1978.

[12] R. Chitnis, N. Talmon, Can we create large k-cores by adding few edges?, in: Proceedings of the 13th International Computer Science Symposium in Russia (CSR), in: Lecture Notes in Computer Science, vol. 10846, Springer, 2018, pp. 78–89.

[13] Z. Zhou, F. Zhang, X. Lin, W. Zhang, C. Chen, K-core maximization: an edge addition approach, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), 2019, pp. 4867–4873.

[14] C. Crespelle, P.G. Drange, F.V. Fomin, P.A. Golovach, A survey of parameterized algorithms and the complexity of edge modification, CoRR, arXiv:2001.06867 [abs].

[15] M.A. Henning, A. Yeo, Tight lower bounds on the matching number in a graph with given maximum degree, J. Graph Theory 89 (2) (2018) 115–149, https://doi.org/10.1002/jgt.22244.

[16] H.W. Lenstra, Integer programming with a fixed number of variables, Math. Oper. Res. 8 (4) (1983) 538–548, https://doi.org/10.1287/moor.8.4.538.

[17] R. Kannan, Minkowski's convex body theorem and integer programming, Math. Oper. Res. 12 (3) (1987) 415–440, https://doi.org/10.1287/moor.12.3.415.

[18] A. Frank, É. Tardos, An application of simultaneous Diophantine approximation in combinatorial optimization, Combinatorica 7 (1) (1987) 49–65, https://doi.org/10.1007/bf02579200.

[19] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015.

[20] P. Erdős, T. Gallai, Gráfok előírt fokszámú pontokkal, Mat. Lapok 11 (1960) 264–274.

[21] F.V. Fomin, D. Lokshtanov, S. Saurabh, M. Zehavi, Kernelization. Theory of Parameterized Preprocessing, Cambridge University Press, 2018.

[22] R. Chitnis, F.V. Fomin, P.A. Golovach, Preventing unraveling in social networks gets harder, in: Proceedings of the 27h AAAI Conference on Artificial Intelligence (AAAI), AAAI Press, 2013, pp. 1085–1091.

[23] R. Chitnis, F.V. Fomin, P.A. Golovach, Parameterized complexity of the anchored k-core problem for directed graphs, Inf. Comput. 247 (2016) 11–22, https://doi.org/10.1016/j.ic.2015.11.002.

[24] F. Zhang, W. Zhang, Y. Zhang, L. Qin, X. Lin, OLAK: an efficient algorithm to prevent unraveling in social networks, Proc. VLDB Endow. 10 (6) (2017) 649–660.

[25] F. Zhang, Y. Zhang, L. Qin, W. Zhang, X. Lin, Finding critical users for social network engagement: the collapsed k-core problem, in: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI), AAAI Press, 2017, pp. 245–251.

[26] J. Luo, H. Molter, O. Suchý, A parameterized complexity view on collapsing k-cores, Theory Comput. Syst. 65 (8) (2021) 1243–1282, https://doi.org/10.1007/s00224-021-10045-w.

[27] F.V. Fomin, P.A. Golovach, F. Panolan, S. Saurabh, Editing to connected f-degree graph, SIAM J. Discrete Math. 33 (2) (2019) 795–836, https://doi.org/10.1137/17M1129428.

[28] P.A. Golovach, Editing to a connected graph of given degrees, Inf. Comput. 256 (2017) 131–147, https://doi.org/10.1016/j.ic.2017.04.013.

[29] P. Goyal, P. Misra, F. Panolan, G. Philip, S. Saurabh, Finding even subgraphs even faster, J. Comput. Syst. Sci. 97 (2018) 1–13, https://doi.org/10.1016/j.jcss.2018.03.001.

[30] L. Mathieson, S. Szeider, Editing graphs to satisfy degree constraints: a parameterized approach, J. Comput. Syst. Sci. 78 (1) (2012) 179–191, https://doi.org/10.1016/j.jcss.2011.02.001.
[31] R. Diestel, Graph Theory, Springer Publishing Company, Incorporated, 2018.
[32] A. Tripathi, S. Vijay, A note on a theorem of Erdős & Gallai, Discrete Math. 265 (1–3) (2003) 417–420, https://doi.org/10.1016/s0012-365x(02)00886-5.
[33] M. Dom, D. Lokshtanov, S. Saurabh, Kernelization lower bounds through colors and ids, ACM Trans. Algorithms 11 (2) (2014) 13, https://doi.org/10.1145/2650261.
[34] H.L. Bodlaender, R.G. Downey, M.R. Fellows, D. Hermelin, On problems without polynomial kernels, J. Comput. Syst. Sci. 75 (8) (2009) 423–434, https://doi.org/10.1016/j.jcss.2009.04.001.
[35] L. Fortnow, R. Santhanam, Infeasibility of instance compression and succinct PCPs for NP, J. Comput. Syst. Sci. 77 (1) (2011) 91–106, https://doi.org/10.1016/j.jcss.2010.06.007.