Building Large k-Cores from Sparse Graphs

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway
fomin@uib.no

Danil Sagunov
St. Petersburg Department of V.A. Steklov Institute of Mathematics, Russia
JetBrains Research, St. Petersburg, Russia
`http://danilka.pro`
danilka.pro@gmail.com

Kirill Simonov
Department of Informatics, University of Bergen, Norway
kirill.simonov@uib.no

Abstract

A popular model to measure network stability is the k-core, that is the maximal induced subgraph in which every vertex has degree at least k. For example, k-cores are commonly used to model the unraveling phenomena in social networks. In this model, users having less than k connections within the network leave it, so the remaining users form exactly the k-core. In this paper we study the question of whether it is possible to make the network more robust by spending only a limited amount of resources on new connections. A mathematical model for the k-core construction problem is the following 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.

The previous studies on Edge k-Core demonstrate that the problem is computationally challenging. In particular, it is NP-hard when $k = 3$, W[1]-hard when parameterized by $k + b + p$ (Chitnis and Talmon, 2018), and APX-hard (Zhou et al, 2019). Nevertheless, we show that there are efficient algorithms with provable guarantee when the k-core has to be constructed from a sparse graph with some additional structural properties. Our results are

- 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. On the other hand, with such parameterization, the problem does not admit a polynomial kernel subject to a widely-believed assumption from complexity theory;
- Edge k-Core is FPT parameterized by the treewidth of the graph plus k. This improves upon a result of Chitnis and Talmon by not requiring b to be small.

Each of our algorithms is built upon a new graph-theoretical result interesting in its own.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases parameterized complexity, k-core, vertex cover, treewidth

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.35

Related Version `https://arxiv.org/abs/2002.07612`

Funding The research leading to these results has received funding from the Research Council of Norway via the project "MULTIVAL" (grant no. 263317).

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 [5, 10, 11], protein function prediction [30], hierarchical structure analysis [3], graph visualization [2], and network clustering and connectivity [1, 19].

In online social networks users tend to contribute content only when a certain amount of their friends do the same [6], 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 [28]: Consider a cycle on n vertices, which is a 2-core with n vertices. Missing just one edge from this graph turns it into in a path and triggers the withdrawals that results 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 withdrawal. We consider the following mathematical model for this problem. For a given a network, the assumption is that a user leaves the network when less than k his/her friends remain within it. We would like to prevent unraveling of the network, so that at least 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 [9] 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 larger number of users [9, 33]. Other potential application 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 vertex degrees, see [12] 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 [9] 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.

[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.

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 [23] 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 [25], see also [24, 18], to solve Edge k-Core. Nowadays ILP is a commonly used tool for designing parameterized algorithms, see e.g. [13, 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. One of the components in the proof of Lemma 13 is the classical Erdős-Gallai theorem [16] about graphic sequences. We complement FPT algorithm by lower bounds on the size of the kernel.

Bounded treewidth. Chitnis and Talmon in [9] 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 the problem is solvable in polynomial time. We enhance this result by proving that Edge k-Core is FPT parameterized by $\mathrm{tw} + k$. As the algorithm of Chitnis and Talmon in [9], 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 criteria how a subset of vertices can be turned into a k-core "optimally". This key insight allows us to show that the problem is FPT parameterized by $\mathrm{tw} + k$.

Related work. The usability of k-cores in the study of network unraveling phenomena was popularized by the influential paper of Bhawalkar et al. [4] 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 [7], where new computational results were obtained and some results of [5] were strengthened. Also, Chitnis, Fomin and Golovach studied this problem applied to networks where the underlying graph is directed [8]. Heuristic algorithms for this problem are discussed in [31].

Edge k-Core was introduced in [9], where also a number of complexity and algorithmic results about the problem were established. Zhou et al. [33] provide some non-approximability results for Edge k-Core as well as some heuristics. The work [32] 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 the work of 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 [12]. 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 [17, 21, 20, 22, 27].

2 Preliminaries

All graphs considered in this paper are simple undirected graphs. We use standard graph notation and terminology, following the book of Diestel [14]. 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.

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 its vertex $v \in V(G)$, let $\mathrm{df}_G(v) = \max\{0, k - \deg_G(v)\}$ denote the* deficiency *of v in G. By $\mathrm{df}(G) = \sum_{v \in V(G)} \mathrm{df}_G(v)$ we denote the* total deficiency *in G.*

Note that an addition of an edge between two vertices of G can decrease $\mathrm{df}(G)$ 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 $\mathrm{df}_G(u) > 0$ and $\mathrm{df}_G(v) > 0$. If $\mathrm{df}_G(u) = 0$ and $\mathrm{df}_G(v) > 0$, 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.

Due to the space restrictions, some of the proofs in this paper are omitted. The results with omitted proofs are marked with the '$\star$' sign. Missing proofs can be found in the full version of this paper.

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-top 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** ($\star$). *EDGE k-CORE is solvable in time $\mathcal{O}(kn^2)$ on the class of forests.*

The algorithm follows a fairly standard technique, so the detailed description of the algorithm and the proof of its correctness are omitted from this extended abstract. Instead for the remaining part of this section we focus on the proof of Theorem 4.

Proof of Theorem 4. The theorem says that completion of T to a graph of total deficiency 0 can be done using $\lceil \frac{1}{2}\,\mathrm{df}(T) \rceil$ edge additions. Note that this bound is tight because a single edge addition decreases the total deficiency by at most two. When $\mathrm{df}(T)$ is even, we have to prove that it is possible to complete T by adding only good edges. When $\mathrm{df}(T)$ is odd, we have to complete T to a graph of total deficiency 1 adding $\lfloor \frac{1}{2}\,\mathrm{df}(T) \rfloor$ 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 $k \geq 4$ 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 $\lfloor \frac{1}{2}\,\mathrm{df}(T) \rfloor$ good edges, in a connected way for $k \geq 4$.

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 $\lfloor \frac{1}{2}\,\mathrm{df}(T) \rfloor$ good edges.

For $k \geq 2$, 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 $k = 1$, pairing the leaves arbitrarily suffices.

Now, for every integer $k \geq 3$, 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 $k \geq 4$ 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 completion is good, thus the completion requires exactly $\frac{1}{2}\,\mathrm{df}(T)$ 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 $(\mathrm{df}(T') \bmod 2)$ using $\lfloor \frac{1}{2}\,\mathrm{df}(T') \rfloor$ 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 $k \geq 4$, 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) $s \neq v$, $t \neq v$ 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'.

$$\text{If } k \geq 4,\ A' \text{ is connected.} \tag{1}$$

The correctness of (1) 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

$$\Delta(A') \leq k - 1. \tag{2}$$

Also

$$|E(A')| \geq \frac{n(k-2)+1}{2}, \tag{3}$$

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'.

$$|V(A')| \geq k. \ \textit{If } n > k+1, \textit{ then } |V(A')| > k. \tag{4}$$

The inequality (4) follows from (2), (3), and the fact that $2 \cdot |E(A')| \le \Delta(A') \cdot |V(A')|$. For the detailed proof, we direct the reader to the full version of this paper.

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 [23].

▶ **Proposition 6** ([23]). *For any integer $t \ge 3$, any connected graph G with $|V(G)| = n$, $|E(G)| = m$ and $\Delta(G) \le t$, contains a matching of size at least*

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

or at least

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

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 $\lceil \frac{k}{2} \rceil$.

The proof is by a careful application of Proposition 6 to (1), (2), (3) and (4), it can be found in the full version of this paper.

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 $(\mathrm{df}(T') \bmod 2)$. That is, $V(G') = V(T')$ and $E(G') = E(T') \sqcup E(A')$. If $\mathrm{df}(T')$ is odd, G' has a single vertex with deficiency one, denote it by u. For every other vertex $s \in V(G')$, $\mathrm{df}_{G'}(s) = 0$. 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 probably 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 $\mathrm{df}(T')$ 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 the rerouting is possible. The case analysis is technical, and we dedicate the details to the full version. For the reference, we list the cases here.

Case (a). *k is even and $p \in V(A')$.*
Case (b). *k is even and $p \notin V(A')$.*
Case (c). *k is odd and $p \notin V(A')$.*
Case (d). *k is odd and $p \in V(A')$, there is no deficient vertex in G'.*
Case (e). *k is odd and $p \in V(A')$, $\mathrm{df}_{G'}(u) = 1$.*

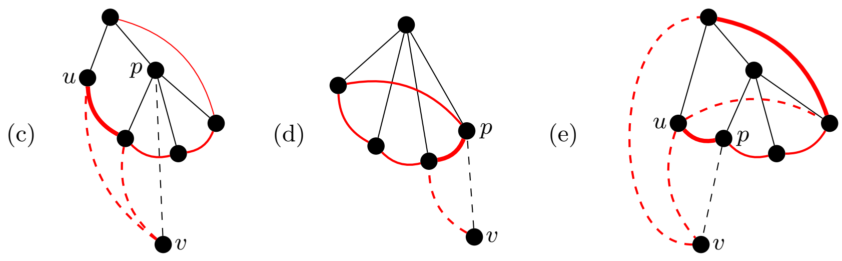

Figure 1 The cases of rerouting for $k = 3$. Solid edges denote the edges of G'. Straight black edges denote the edges of T', and curved red edges denote the edges of A'. Edges of the matching in A' that are deleted in G are highlighted bold. Dashed edges denote the newly-added edges in G.

Clarifying pictures for cases (c), (d) and (e), corresponding to odd k, are presented in Figure 1. Considering each case required to accomplish the inductive step 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 [9], 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** ([9]). *EDGE k-CORE is* NP*-hard even on the class of* 2*-degenerate graphs for $k = 3$.*

4 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 an instance of integer linear program (ILP) whose 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 [13, 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 [29] of the classical theorem of Erdős and Gallai about degree sequences [16].

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 [25].

▶ **Proposition 9** ([24, 25, 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. [13]. 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 to 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 to 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 $d_1, d_2, \dots, d_n$ of n non-negative integers. is called* graphic *if there exists a graph G with $V(G) = \{v_1, v_2, \dots, v_n\}$, 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 [16].

▶ **Proposition 12** (Erdős-Gallai Theorem, [16]). *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 [29].

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

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. Its formal proof is given in the full version of the paper and accumulates 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.*

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 [15] that the Bounded Rank Disjoint Sets problem does not admit a polynomial kernel, and our proof is by reduction from this problem.

▶ **Theorem 18** (⋆). *Unless* NP ⊆ coNP/poly, *Edge k-Core does not admit a polynomial kernel when parameterized by the combined parameter $\mathrm{vc} + k + b + p$.*

5 Treewidth

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

▶ **Proposition 19** ([9]). *Edge k-Core can be solved in time $(k + \mathrm{tw})^{\mathcal{O}(\mathrm{tw}+b)} \cdot n^{\mathcal{O}(1)}$.*

We start with the central combinatorial result of this section which allows the algorithmic improvement. Namely, we show that whenever the total deficiency of a graph G exceeds a polynomial in k, G can be completed to a graph of minimum degree k using the minimum possible number of edges. 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 $k \geq 2$, any graph G with $\operatorname{df}(G) \geq 3k^3$ can be completed to a graph of minimum degree k using $\lceil \frac{1}{2} \operatorname{df}(G) \rceil$ edge additions with a polynomial-time algorithm.*

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

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

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

Now we exhaustively apply the new rule. Fix two vertices $u, v \in C$, such that either u and v are distinct, or $u = v$ and $\operatorname{df}_{G'}(u) \geq 2$. 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 new rule is no longer applicable, the size of C is at most one, and $\operatorname{df}(G')$ 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 $|C| \leq k$, 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 $|C| \leq k$ 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, $\operatorname{df}(G) = 2|B| + \operatorname{df}(G')$, and $\operatorname{df}(G') \leq |C| \cdot k \leq k^2$. So $\operatorname{df}(G) < 3k^3$ contradicting the statement.

Therefore, by the constructed sequence of good additions we reached the situation when $|C|$ and $\operatorname{df}(G')$ 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 $\operatorname{df}(G) = 2|B| + \operatorname{df}(G')$, $\operatorname{df}(G)$ 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 $\operatorname{df}(G) \geq 3k^3$. ◀

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 $b \geq 3k^3$ 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.

▶ **Lemma 22** ($\star$). *Given an n-vertex graph G of treewidth tw and integers k, p, the value*

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

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

All pieces together give the main algorithmic result of this section.

▶ **Theorem 23** (⋆). *Edge k-Core admits an* FPT *algorithm when parameterized by the combined parameter* $\text{tw} + k$.

References

1 J. Ignacio Alvarez-Hamelin, Mariano G. Beiró, and Jorge Rodolfo Busch. Understanding edge connectivity in the internet through core decomposition. *Internet Mathematics*, 7(1):45–66, 2011. doi:10.1080/15427951.2011.560786.

2 J. Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in Neural Information Processing Systems 18 (NIPS)*, pages 41–50, 2005.

3 José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. K-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases. *Networks & Heterogeneous Media*, 3(2):371–393, 2008.

4 Kshipra Bhawalkar, Jon M. Kleinberg, Kevin Lewi, Tim Roughgarden, and Aneesh Sharma. Preventing unraveling in social networks: The anchored k-core problem. In *ICALP '12*, volume 7392 of *Lecture Notes in Computer Science*, pages 440–451, 2012. doi:10.1007/978-3-642-31585-5_40.

5 Kshipra Bhawalkar, Jon M. Kleinberg, Kevin Lewi, Tim Roughgarden, and Aneesh Sharma. Preventing unraveling in social networks: The anchored k-core problem. *SIAM J. Discrete Math.*, 29(3):1452–1475, 2015. doi:10.1137/14097032X.

6 Moira Burke, Cameron Marlow, and Thomas 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)*, pages 945–954. ACM, 2009. doi:10.1145/1518701.1518847.

7 Rajesh Chitnis, Fedor V Fomin, and Petr A Golovach. Preventing unraveling in social networks gets harder. In *Proceedings of the 27h AAAI Conference on Artificial Intelligence (AAAI)*, pages 1085–1091. AAAI Press, 2013.

8 Rajesh Chitnis, Fedor V. Fomin, and Petr A. Golovach. Parameterized complexity of the anchored k-core problem for directed graphs. *Inf. Comput.*, 247:11–22, 2016. doi:10.1016/j.ic.2015.11.002.

9 Rajesh Chitnis and Nimrod Talmon. Can we create large k-cores by adding few edges? In *Proceedings of the 13th International Computer Science Symposium in Russia (CSR)*, volume 10846 of *Lecture Notes in Computer Science*, pages 78–89. Springer, 2018. doi:10.1007/978-3-319-90530-3_8.

10 M.S.Y. Chwe. Structure and Strategy in Collective Action 1. *American Journal of Sociology*, 105(1):128–156, 1999.

11 M.S.Y. Chwe. Communication and Coordination in Social Networks. *The Review of Economic Studies*, 67(1):1–16, 2000.

12 Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. *CoRR*, abs/2001.06867, 2020. arXiv:2001.06867.

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

14 Reinhard Diestel. *Graph theory*. Springer Publishing Company, Incorporated, 2018.

15 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and IDs. *ACM Transactions on Algorithms*, 11(2):1–20, 2014.

16 Paul Erdős and Tibor Gallai. Gráfok előírt fokszámú pontokkal. *Matematikai Lapok*, 11:264–274, 1960.

17 Fedor V. Fomin, Petr A. Golovach, Fahad Panolan, and Saket Saurabh. Editing to connected f-degree graph. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 47 of *LIPIcs*, pages 36:1–36:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. `doi:10.4230/LIPIcs.STACS.2016.36`.

18 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, March 1987. `doi:10.1007/bf02579200`.

19 Christos Giatsidis, Fragkiskos Malliaros, Dimitrios Thilikos, and Michalis Vazirgiannis. Corecluster: A degeneracy based graph clustering framework. In *Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI).* AAAI Press, 2014.

20 Petr A. Golovach. Editing to a graph of given degrees. In *Proceedings of the 9th International Symposium Parameterized and Exact Computation (IPEC)*, volume 8894 of *Lecture Notes in Comput. Sci.*, pages 196–207. Springer, 2014.

21 Petr A. Golovach. Editing to a connected graph of given degrees. *Inf. Comput.*, 256:131–147, 2017.

22 Prachi Goyal, Pranabendu Misra, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. Finding even subgraphs even faster. *J. Comput. Syst. Sci.*, 97:1–13, 2018.

23 Michael A. Henning and Anders Yeo. Tight lower bounds on the matching number in a graph with given maximum degree. *Journal of Graph Theory*, 89(2):115–149, 2018. `doi:10.1002/jgt.22244`.

24 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, August 1987. `doi:10.1287/moor.12.3.415`.

25 H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, November 1983. `doi:10.1287/moor.8.4.538`.

26 Junjie Luo, Hendrik Molter, and Ondřej Suchý. A parameterized complexity view on collapsing k-cores. In *13th International Symposium on Parameterized and Exact Computation*, 2019.

27 Luke Mathieson and Stefan Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. *J. Comput. Syst. Sci.*, 78(1):179–191, 2012.

28 T.C. Schelling. *Micromotives and Macrobehavior.* WW Norton, 2006.

29 Amitabha Tripathi and Sujith Vijay. A note on a theorem of erdős & gallai. *Discrete Mathematics*, 265(1-3):417–420, April 2003. `doi:10.1016/s0012-365x(02)00886-5`.

30 Stefan Wuchty and Eivind Almaas. Peeling the yeast protein network. *Proteomics*, 5(2):444–449, 2005.

31 Fan Zhang, Wenjie Zhang, Ying Zhang, Lu Qin, and Xuemin Lin. OLAK: an efficient algorithm to prevent unraveling in social networks. *Proceedings of the VLDB Endowment*, 10(6):649–660, 2017.

32 Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. Finding critical users for social network engagement: The collapsed k-core problem. In *Procedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 245–251. AAAI Press, 2017.

33 Zhongxin Zhou, Fan Zhang, Xuemin Lin, Wenjie Zhang, and Chen Chen. K-core maximization: An edge addition approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4867–4873. ijcai.org, 2019. `doi:10.24963/ijcai.2019/676`.