Theory of Computing Systems (2021) 65:1243–1282
https://doi.org/10.1007/s00224-021-10045-w

A Parameterized Complexity View on Collapsing *k*-Cores

Junjie Luo[1,2,3] · **Hendrik Molter**[1] · **Ondřej Suchý**[4]

Accepted: 17 April 2021 / Published online: 19 June 2021

Abstract

We study the NP-hard graph problem COLLAPSED k-CORE where, given an undirected graph G and integers b, x, and k, we are asked to remove b vertices such that the k-core of remaining graph, that is, the (uniquely determined) largest induced subgraph with minimum degree k, has size at most x. COLLAPSED k-CORE was introduced by Zhang et al. (2017) and it is motivated by the study of engagement behavior of users in a social network and measuring the resilience of a network against user drop outs. COLLAPSED k-CORE is a generalization of r-DEGENERATE VERTEX DELETION (which is known to be NP-hard for all $r \geq 0$) where, given an undirected graph G and integers b and r, we are asked to remove b vertices such that the remaining graph is r-degenerate, that is, every its subgraph has minimum degree at most r. We investigate the parameterized complexity of COLLAPSED k-CORE with respect to the parameters b, x, and k, and several structural parameters of the input graph. We reveal a dichotomy in the computational complexity of COLLAPSED k-CORE for $k \leq 2$ and $k \geq 3$. For the latter case it is known that for all $x \geq 0$ COLLAPSED k-CORE is W[P]-hard when parameterized by b. For $k \leq 2$ we show that COLLAPSED k-CORE is W[1]-hard when parameterized by b and in FPT when parameterized by $(b + x)$. Furthermore, we outline that COLLAPSED k-CORE is in FPT when parameterized by the treewidth of the input graph and presumably does not admit a polynomial kernel when parameterized by the vertex cover number of the input graph.

Junjie Luo was partially supported by CAS-DAAD Joint Fellowship Program for Doctoral Students of UCAS and partially supported by the DFG, project AFFA (NI 369/15 and BR 5207/1). Hendrik Molter was supported by the DFG, project MATE (NI 369/17). Ondřej Suchý was supported by grant 17-20065S of the Czech Science Foundation.

An extended abstract of this work appears in the proceedings of the 13th International Symposium on Parameterized and Exact Computation (IPEC 2018) [36].

✉ Junjie Luo
junjie.luo@campus.tu-berlin.de; luojunjie@amss.ac.cn

Extended author information available on the last page of the article.

Keywords r-Degenerate vertex deletion · Feedback vertex set · Fixed-parameter tractability · Kernelization lower bounds · Graph algorithms · Social network analysis

1 Introduction

In recent years, modelling user engagement in social networks has received substantial interest [44, 45]. A popular assumption is that a user engages in a social network platform if she has at least a certain number of contacts, say k, on the platform. Further, she is inclined to abandon the social network if she has less than k contacts [6, 15, 16, 25, 37, 46]. In compliance with this assumption, a suitable graph-theoretic model for the "stable" part of a social network is the so-called *k-core* of the social network graph, that is, the largest induced subgraph with minimum degree k [42].[1]

Now, given a stable social network, that is, a graph with minimum degree k, the departure of a user decreases the degree of her neighbors in the graph by one which then might be smaller than k for some of them. Following our assumption these users now will abandon the network, too. This causes a cascading effect of users dropping out (collapse) of the network until a new stable state is reached. From an adversarial perspective a natural question is how to maximally destabilize a competing social network platform by compelling b users to abandon the network. This problem was introduced as Collapsed k-Core by Zhang et al. [46] and the decision version is formally defined as follows.

Collapsed k-Core

Input: An undirected graph $G = (V, E)$, and integers b, x, and k.

Question: Is there a set $S \subseteq V$ with $|S| \leq b$ such that the k-core of $G - S$ has size at most x?

In the mentioned motivation, one would aim to minimize x for a given b and k. Alternatively, we can also interpret this problem as a measure for resilience agains user drop outs of a social network by determining the smallest b for a given k and x.

Related Work In 2017 Zhang et al. [46] showed that Collapsed k-Core is NP-hard for any $k \geq 1$ and gave a greedy algorithm to compute suboptimal solutions for the problem. However, for $x = 0$ and any fixed $k \geq 1$, solving Collapsed k-Core is equivalent to finding b vertices such that after removing said vertices, the remaining graph is $(k-1)$-degenerate[2]. This problem is known as r-Degenerate Vertex Deletion and it is defined as follows.

r-Degenerate Vertex Deletion

Input: An undirected graph $G = (V, E)$, and integers b and r.

Question: Is there a set $S \subseteq V$ with $|S| \leq b$ such that $G - S$ is r-degenerate?

It is easy to see that Collapsed k-Core is a generalization of r-Degenerate Vertex Deletion. In 2010 Mathieson [38] showed that r-Degenerate Vertex

[1] Note that the k-core of a graph is uniquely determined.

[2] A graph G is r-degenerate if every subgraph of G has a vertex with degree at most r [21].

DELETION is NP-complete and W[P]-complete when parameterized by the budget b for all $r \geq 2$ even if the input graph is already $(r+1)$-degenerate and has maximum degree $2r+1$. In the mid-90s Abrahamson et al. [1] already claimed W[P]-completeness for r-DEGENERATE VERTEX DELETION with $r = 2$ when parameterized by b under the name DEGREE 3 SUBGRAPH ANNIHILATOR.

For $r = 1$, the problem is equivalent to FEEDBACK VERTEX SET and for $r = 0$ it is equivalent to VERTEX COVER, both of which are known to be NP-complete and fixed-parameter tractable when parameterized by the solution size [20, 26]. For VERTEX COVER already the first known FPT algorithm is deterministic single exponential linear time and the currently fastest algorithm runs in $O(1.2738^b + bn)$ time [11]. For FEEDBACK VERTEX SET randomized algorithms with running time of such kind were known [19], whereas the running time of single exponential linear time deterministic algorithms was unsatisfactory [27, 35]. This changed with the linear time computable polynomial kernel of Iwata [29] which, combined with some previously known single exponential algorithms such as the $O(3.619^b \cdot n^{O(1)})$ algorithm of [33], produces efficient algorithms of such kind. Very recently native deterministic single exponential linear time algorithms appeared [9], achieving $O(3.460^b \cdot n)$ running time [30]. The aforementioned results concerning r-DEGENERATE VERTEX DELETION in fact imply the hardness results shown by Zhang et al. [46] for COLLAPSED k-CORE.

Chitnis and Talmon [14] recently studied the EDGE k-CORE problem: given a graph G, a budget b, and a goal p, can at most b edges be added to G to obtain a k-core containing at least p vertices? They showed the problem to be polynomial time solvable for $k \leq 2$, while NP-complete for $k \geq 3$. Furthermore, they showed that the problem is W[1]-hard with respect to $(b + k + p)$ and designed an algorithm with running time $(k+\mathrm{tw})^{O(\mathrm{tw}+b)} \cdot \mathrm{poly}(n)$, where tw is the treewidth of the input graph G.

COLLAPSED k-CORE with $x = 0$ (or r-DEGENERATE VERTEX DELETION) can be also viewed as a special case of the TARGET SET SELECTION problem introduced by Kempe et al. [31]. Here one is given a graph G, a budget b, and a threshold function $f : V(G) \to \mathbb{N}$. A vertex v gets activated once at least $f(v)$ of its neighbors are activated. The question is whether there is an initial set (target set) of at most b vertices such that, if we initially activate this set, then eventually all vertices of the graph get activated. COLLAPSED k-CORE with $x = 0$ corresponds to the case that the threshold function $f(v)$ equals $\deg(v) - k + 1$ for every vertex v.

TARGET SET SELECTION is known to be NP-hard even for split graphs of diameter 2 [40] and the minimum size of a target set is APX-hard to approximate [12] and also resistant to parameterized approximation [4]. It is W[1]-hard with respect to feedback vertex set (and treewidth) [5], W[2]-hard with respect to budget b on split graphs [40], and FPT with respect to cluster editing number, vertex cover number, and feedback edge set number [40]. More tractability results can be obtained if the thresholds are bounded by a constant [5, 17, 28] or equal to half the degree of the vertex (majority constraint) [23]. We refer the reader to some of the papers [17, 40] for a more detailed survey of the results. A concept similar to TARGET SET SELECTION with the majority constraint is studied under the names "local influence", "majority consensus", "opinion forming", etc. [41].

A somewhat dual problem to TARGET SET SELECTION is HARMLESS SET, where given the same input, the question is to find a set of *at least* b vertices in which each vertex has less than $f(v)$ neighbors [3].

Our Contribution We complete the parameterized complexity landscape of COLLAPSED k-CORE with respect to the parameters b, k, and x. Specifically, we correct errors in the literature [1, 38] concerning the W[P]-completeness of r-DEGENERATE VERTEX DELETION when parameterized by b for $r = 2$. We clarify the parameterized complexity of COLLAPSED k-CORE for $k \leq 2$ by showing W[1]-hardness for parameter b and fixed-parameter tractability for the combination of b and x. Together with previously known results, this reveals a dichotomy in the computational complexity of COLLAPSED k-CORE for $k \leq 2$ and $k \geq 3$.

We present two single exponential linear time FPT algorithms, one for COLLAPSED k-CORE with $k = 1$ and one for $k = 2$. In both cases the parameter is $(b+x)$. In particular, the algorithm for $k = 2$ runs in $O(1.755^{x+4b} \cdot n)$ time which means that it solves FEEDBACK VERTEX SET in $O(9.487^{b} \cdot n)$ time (here, b is the solution size of FEEDBACK VERTEX SET). Cao [9] independently developed a very similar algorithm for FEEDBACK VERTEX SET. In comparison, we additionally generalize the algorithm for COLLAPSED k-CORE and give a more thorough running time analysis. A recent, even more thorough, analysis of an algorithm similar to Cao's algorithm proves it to be the fastest known deterministic algorithm for FEEDBACK VERTEX SET [30].

Furthermore, we conduct a thorough parameterized complexity analysis with respect to structural parameters of the input graph. On the positive side, we show that COLLAPSED k-CORE is fixed-parameter tractable when parameterized by the treewidth of the input graph and show that it presumably does not admit a polynomial kernel when parameterized by either the vertex cover number or the bandwidth of the input graph. We also show that the problem is fixed-parameter tractable when parameterized by the combination of the cliquewidth of the input graph and b or x. Further results include W[1]-hardness when parameterized by the clique cover number of the input graph and para-NP-hardness for the domination number of the input graph.

2 Hardness Results from the Literature

In this section, we gather and discuss known hardness results for COLLAPSED k-CORE. Recall that COLLAPSED k-CORE with $x = 0$ is the same problem as r-DEGENERATE VERTEX DELETION with $r = k - 1$. Hence, the hardness of COLLAPSED k-CORE was first established by Mathieson [38] who showed that r-DEGENERATE VERTEX DELETION is NP-complete and W[P]-complete when parameterized by the budget b for all $r \geq 2$ even if the input graph is already $(r+1)$-degenerate and has maximum degree $2r+1$. However, in the proof of Mathieson [38] the reduction is incorrect for the case $r = 2$. Abrahamson et al. [1] claim W[P]-completeness for $r = 2$ but their reduction is also flawed. We provide

counterexamples for both cases and show how to adjust the reduction of Mathieson [38].

2.1 Counterexamples

Counterexample for Mathieson's Reduction [38]**.** We refer to the original paper by Mathieson [38] for definitions, notation, and description of the gadgets and the reduction itself. Mathieson provides a reduction from CYCLIC MONOTONE CIRCUIT ACTIVATION to r-DEGENERATE VERTEX DELETION [38, Theorem 4.4] showing that r-DEGENERATE VERTEX DELETION is W[P]-complete when parameterized by the budget b for all $r \geq 2$ even if the input graph is already $(r+1)$-degenerate and has maximum degree $2r+1$. However, for the case of $r = 2$ it is easy to see that the OR gadget is already 2-degenerate. We illustrate the flawed OR gadget for $r = 2$ in Fig. 1.

This means that whenever a CYCLIC MONOTONE CIRCUIT ACTIVATION instance is activated by the set of all its binary OR gates, the graph produced by the reduction (for $r = 2$) is already 2-degenerate, which clearly makes the reduction incorrect. We give an example in Fig. 2.

We describe how to repair the reduction (for $r = 2$) in the proof of Theorem 1.

Counterexample for the Reduction of Abrahamson et al. [1] We refer to the original paper by Abrahamson et al. [1] for definitions, notation, and description of the gadgets and the reduction itself. The same reduction (using the same notation) can

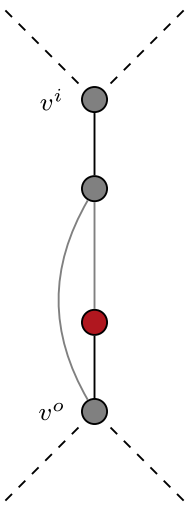

Fig. 1 Illustration of the OR gadget in the reduction of Mathieson [38, Theorem 4.4] for $r = 2$. Note that the red colored vertex has degree 2 and the whole gadget is 2-degenerate

also be found in the book "Fundamentals of Parameterized Complexity" by Downey and Fellows [22]. Abrahamson et al. provide a reduction from WEIGHTED MONOTONE CIRCUIT SATISFIABILITY to DEGREE 3 SUBGRAPH ANNIHILATOR, which is equivalent to r-DEGENERATE VERTEX DELETION with $r = 2$. With this reduction they claim to show that r-DEGENERATE VERTEX DELETION with $r = 2$ is W[P]-complete when parameterized by the budget b [1, Theorem 3.7 (ii)].

The main idea of their reduction is that once a satisfying assignment is found, all variable gadgets corresponding to variables that are set to false are also removed. However, since in a variable gadget for a variable $x[i]$, the vertex $v(i, 4)$ has high degree, it is only removed if sufficiently many of the gate gadgets that it is connected to are also removed. However, an AND gadget combined with a fan-out gadget is only removed if both inputs are removed. This follows from fan-out gadget not being removable from below. This allows us to create a counterexample with $b = 1$, which we illustrate in Fig. 3. It is easy to check that $x[1] = x[2] =$ false, $x[3] =$ true is the only satisfying assignment that has at most one variable set to true. Furthermore, the AND gates in the red area have two outgoing connections each, hence the corresponding AND gadgets have fan-out gadgets attached to them. Initially, the output of these gates is false for the satisfying assignment so the fan-out gadgets attached to them are only removed if the AND gadgets themselves are removed. An AND gadget is removed if both of its inputs are removed. However, note that the variable gadget for $x[1]$ is not completely removed after $v(3, 4)$ is removed from the graph. In particular, the vertex $v(1, 4)$ has still degree $m(b + 1) > 3$ (where m is the maximum fan-out of all variables) after all vertices are removed since the AND gadgets it connects to are not removed. It follows that the graph is not 2-degenerate after removing $v(3, 4)$ and hence the reduction is not correct.

We believe that the reduction can be corrected by replacing the high degree vertices $v(i, 4)$ by fan-out gadgets that connect to the gate gadgets. However, we omit a proof of this claim.

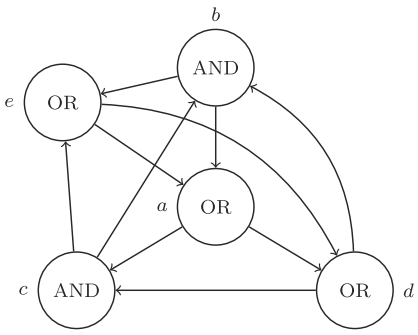

Fig. 2 An example instance of CYCLIC MONOTONE CIRCUIT ACTIVATION. The sets $\{a\}$, $\{b\}$, $\{e\}$, $\{c, d\}$, and their supersets activate the entire circuit, whereas the sets $\{c\}$ and $\{d\}$ do not. In particular, the set $\{a, d, e\}$ of all OR gates activates the entire circuit

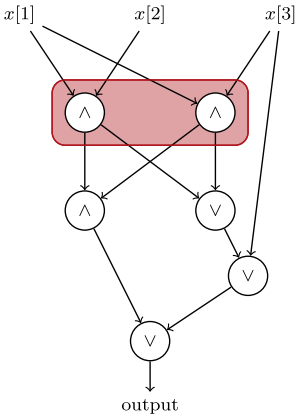

Fig. 3 A WEIGHTED MONOTONE CIRUIT SATISFIABILITY instance with $k = 1$ (max. number of variables set to true). It is easy to check that $x[1] = x[2] =$ false, $x[3] =$ true is the only satisfying assignment that has at most one variable set to true. The AND gates in the red area have two outgoing connections each, hence the corresponding AND gadgets have fan-out gadgets attached to them

2.2 Corrected Proof and Corollaries

Theorem 1 (Corrected from [38]) *For any $r \geq 2$ r-*DEGENERATE VERTEX DELETION *is NP-hard and W[P]-complete when parameterized by b, even if the degeneracy of the input graph is $r + 1$ and the maximum degree of the input graph is $2r + 1$.*

Proof Mathieson provides a reduction from CYCLIC MONOTONE CIRCUIT ACTIVATION to r-DEGENERATE VERTEX DELETION [38, Theorem 4.4] showing that r-DEGENERATE VERTEX DELETION is W[P]-complete when parameterized by the budget b for all $r \geq 2$ even if the input graph is already $(r + 1)$-degenerate and has maximum degree $2r + 1$.

While the reduction is incorrect for $r = 2$ as shown above, there is a way to correct it: We refer to the original paper by Mathieson [38] for definitions, notation, and description of the gadgets and the reduction itself. The only problem is that his OR gadget is 2-degenerate, so the graph sometimes collapses without even deleting a single vertex. Before we introduce the correct gadget, note that the gadget is always used with exactly two inputs (predecessor gates). We can replace the OR gadget for case $r = 2$ with the graph illustrated in Fig. 4.

To collapse this gadget one can simply delete v^o. The correctness now follows from an analogous argument as given by Mathieson [38]. □

The following observation shows that the hardness result by Mathieson [38] (Theorem 1) easily transfers to COLLAPSED k-CORE (also in the cases where $x \neq 0$).

Observation 1 *Let $x' > 0$ be a positive integer. There is a linear time reduction which transforms instances (G, b, x, k) of* COLLAPSED k-CORE *with $x = 0$ into equivalent instances (G', b, x', k) of* COLLAPSED k-CORE. *Moreover, the degeneracy of G' is at most $\max\{d, x'-1\}$ if $x' > k$ and d if $x' \leq k$, where d is the degeneracy of G.*

Proof We distinguish two cases, depending on the relation of x' to k.

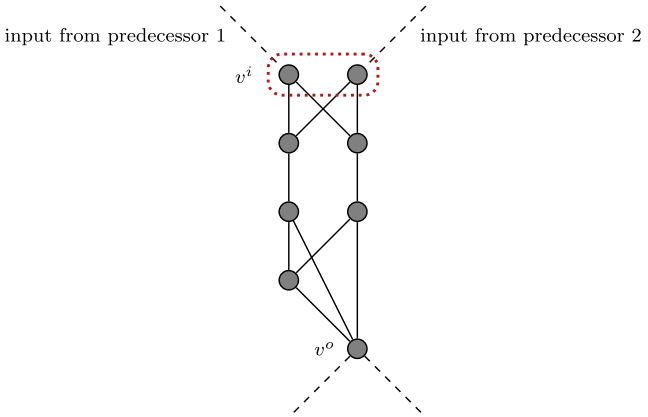

Fig. 4 Illustration of the corrected OR gadget for $r = 2$ in the proof of Theorem 1. The two vertices surrounded by the red dotted line play the role of v^i in the original version of the gadget

If $x' \leq k$, then we let $G' = G$. Obviously, if S is a solution for (G, b, x, k), then it is also a solution for (G', b, x', k). On the other hand, if S' is a solution for (G', b, x', k), then $G' \setminus S' = G \setminus S'$ has a k-core with at most k vertices. However, any vertex of a graph with at most k vertices has degree at most $k-1$ and, thus, the k-core is empty. Therefore S' is a solution for (G, b, x, k). As $G' = G$, the bound on degeneracy follows for this case.

If $x' \geq k+1$, then we obtain G' as a disjoint union of G and a clique C on x' vertices. As the degeneracy of C is $x'-1$, the bound on degeneracy of G' follows for this case. Again obviously, if S is a solution for (G, b, x, k), then it is also a solution for (G', b, x', k), since the k-core of $G' \setminus S$ is exactly C.

Now let S' be a solution for (G', b, x', k). Note that if one vertex of $C \setminus S'$ is part of the k-core of $G' \setminus S'$, then all vertices of $C \setminus S'$ are. If indeed $C \setminus S'$ is a part of the k-core of $G' \setminus S'$, then the k-core contains at most $|C \cap S'|$ other vertices. If X is the set of vertices in the k-core of $G' \setminus (S' \cup C) = G \setminus S'$, then $X \cup (S' \setminus C)$ is a solution for (G, b, x, k).

Now if $C \setminus S'$ is not a part of the k-core of $G \setminus S'$, then we know that $|C \cap S'|$ vertices are sufficient to collapse a clique of size x'. Since the k-core of $G' \setminus S'$, which is the same as the k-core of $G \setminus S'$ is of size at most x', there is a set B of at most $|C \cap S'|$ vertices such that the k-core of $G \setminus (S' \cup B)$ is empty. Hence $((S' \setminus C) \cup B)$ is a solution for (G, b, x, k). □

With that, we arrive at the following corollary.

Corollary 1 Collapsed k-Core *is NP-hard and W[P]-hard when parameterized by b for all $x \geq 0$ and $k \geq 3$, even if the degeneracy of the input graph is* $\max\{k, x-1\}$ *and the maximum degree of the input graph is* $\max\{2k-1, x-1\}$.

Note that r-DEGENERATE VERTEX DELETION is known to be NP-hard for all $r \geq 0$.[3] Hence, we also know that COLLAPSED k-CORE is NP-hard for $k \leq 2$ and all $x \geq 0$. However, the parameterized complexity with respect to b is open in this case. We settle this in the next section.

3 Algorithms and Complexity for $k = 1$ and $k = 2$

In this section we investigate the parameterized complexity of COLLAPSED k-CORE for the case that $k \leq 2$. Since Corollary 1 only applies for $k \geq 3$ we first show in the following that the problem is W[1]-hard with respect to the combination of b and $(n - x)$ for all $k \geq 1$. Furthermore, we present two algorithms; one that solves COLLAPSED k-CORE with $k = 1$ and one for the $k = 2$ case. Both algorithms run in single exponential linear FPT-time with respect to the parameter combination $(b+x)$.

3.1 W[1]-hardness with Respect to b and $n - x$

We first give a parameterized reduction from CLIQUE to COLLAPSED k-CORE. Note that since this hardness result holds for the combination of b and the dual parameter of x, it is incomparable to Corollary 1 even for $k \geq 3$.

Proposition 1 COLLAPSED k-CORE *is W[1]-hard when parameterized by the combination of b and $(n - x)$ for all $k \geq 1$, even if the input graph is bipartite and* $\max\{2, k\}$*-degenerate.*

Proof We reduce from W[1]-hard problem CLIQUE [20], where given a graph $G = (V, E)$ and an integer p, the task is to decide whether G contains a clique of size at least p. Let (G, p) be an instance of CLIQUE and k be a given constant. We build an instance (G', b, x, k) of COLLAPSED k-CORE as follows. We can assume that $p \leq |V(G)|$, as otherwise we can output a trivial no-instance. We further assume that each vertex of G has degree at least $p + 1$. A vertex of degree less than $p - 1$ is not part of a clique of size at least p, while for all vertices of degree $p - 1$ or p we can check in $O(|V(G)| \cdot p^3)$ time whether there is a clique of size at least p containing any of them. Similarly, we assume that $p \geq 4$, so that $p < \binom{p}{2}$ as otherwise we can find the answer in $O(|V(G)|^3)$ time.

We let $V(G') = V \cup U \cup W$, where $V = V(G)$ are the vertices of G, $U = \{u_e^i \mid e \in E, i \in \{1, \ldots, k\}\}$, and $W = \{w_e^i \mid e \in E, i \in \{1, \ldots, k-1\}\}$. We also let $E(G') = E_U \cup E_W$, where $E_U = \{\{v, u_e^i\} \mid e \in E, i \in \{1, \ldots, k\}, v \in e\}$, and $E_W = \{\{u_e^i, w_e^{i'}\} \mid e \in E, i \in \{1, \ldots, k\}, i' \in \{1, \ldots, k-1\}\}$. We actually only introduce the sets W and E_W if $k \geq 2$.

[3] Theorem 1 states NP-hardness of r-DEGENERATE VERTEX DELETION for $r \geq 2$. Recall that for $r = 1$ r-DEGENERATE VERTEX DELETION is equivalent to FEEDBACK VERTEX SET and for $r = 0$ it is equivalent to VERTEX COVER, both of which are known to be NP-hard [26].

Finally we set $b = p$ and $x = n' - (p + (2k-1)\binom{p}{2})$, where $n' = |V(G')|$ is the number of vertices of graph G'.

We claim that (G', b, x, k) is a yes-instance of COLLAPSED k-CORE if and only if (G, p) is a yes-instance of CLIQUE.

$\Rightarrow$: If S is a clique of size at least p in G, then we claim that deleting S from G' results in a k-core of size at most x. Let e be an edge between two vertices of S in G. Then for any $i \in \{1, \ldots, k\}$ vertex u_e^i has only vertices $w_e^1, \ldots w_e^{k-1}$ as neighbors in $G' \setminus S$. Hence, it has degree $k-1$ in this graph and it is not part of the k-core of the graph. Since this holds for each i, vertices $w_e^1, \ldots w_e^{k-1}$ do not have any neighbors in the k-core of $G' \setminus S$ and, hence, they are also not in the k-core of the graph. This makes $2k-1$ vertices per each edge of the clique which are not deleted and not in the k-core, showing that the size of the k-core is at most x.

$\Leftarrow$: Now let S be a set of vertices of $V(G')$ of size at most b such that $G' \setminus S$ has k-core of size at most x. For $e \in E$ let $UW_e = \{u_e^i \mid i \in \{1, \ldots, k\}\} \cup \{w_e^i \mid i \in \{1, \ldots, k-1\}\}$. Let $S_E = \{e \in E \mid S \cap UW_e \neq \emptyset\}$. Note that if $e = \{x, y\}$, $e \notin S_E$, and $|S \cap \{x, y\}| \leq 1$, then the whole set $UW_e \cup \{x, y\} \setminus S$ is in the k-core of $G' \setminus S$ as each vertex in $UW_e \cup \{x, y\} \setminus S$ has at least k neighbors in $UW_e \cup \{x, y\} \setminus S$. This means that each vertex in $V \setminus S$ is in the k-core of $G' \setminus S$. Indeed, for an arbitrary vertex v in $V \setminus S$ the degree of v in G is at least $p+1$ and, thus, there is at least one edge e incident to v which is not in S_E. Hence v is in the k-core of $G' \setminus S$ by the above argument.

For $e \in S_E$ possibly no vertex of UW_e is in the k-core of $G' \setminus S$, effectively shrinking it by $2k-1$ vertices. However, this does not influence the other vertices in V, U, or W, since $V \setminus S$ is in the k-core, as we already observed. If $e = \{x, y\}$ and $\{x, y\} \subseteq S$, then also the whole set UW_e is not in the k-core as observed in the first implication. Thus, if $|S \cap V| = a$ and $|S_E| = c$, then there are at most $(2k-1)(\binom{a}{2} + c)$ vertices of G' which are neither in S nor in the k-core of $G' \setminus S$. As S is a solution, this number has to be at least $(2k-1)\binom{p}{2}$, while $a + c \leq b = p$. It follows that $a = p$ and S is a clique of size p in G.

Note that graph G' is bipartite and for $k \geq 2$ it is also k-degenerate, as all vertices in W have degree k, after removing them the vertices of U have degree 2, and, finally, V forms an independent set in G'. □

3.2 Algorithm for $k = 1$

Now we proceed with the algorithm for COLLAPSED k-CORE with $k = 1$. While there is a simple algorithm with $O(3^{x+b}(m+n))$ running time[4] for this case, we consider an algorithm with the slightly worse running time as stated, since we then generalize this algorithm to the case $k = 2$ with some modifications.

[4] An informal description of the algorithm: We use an initially empty set X that should contain vertices of the remaining 1-core. We branch over edges where both endpoints are not in X and either remove one of the endpoints or put both endpoint into X. Then we branch over all edges that have exactly one endpoint in X and either remove the other endpoint or put the other endpoint into X as well.

Proposition 2 COLLAPSED k-CORE *with* $k = 1$ *can be solved in* $O(2^{x+2b}(m + n))$ *time and in* $O(2^{O(b+\sqrt{bx})} \cdot n)$ *time. Assuming the Exponential Time Hypothesis, there is no* $2^{o(b)}n^{f(x)}$ *time algorithm for* COLLAPSED k-CORE *with* $k = 1$*, for any function* f.

Algorithm 1 Algorithm for COLLAPSED k-CORE with $k = 1$.

```
1  SolveRec(G, S, Q, b, x)
2  if |S| > b or |Q| > b + x then return No solution;
3  Let G' be the 1-core of G \ S.
4  if |V(G')| ≤ x then return S;
5  if |S| = b then return No solution;
6  if V(G') ⊆ Q then return No solution;
7  Let v be a vertex with the highest degree in G' which is not in Q
8  T ← SolveRec(G, S ∪ {v}, Q, b, x)
9  if T ≠ No solution then return T ;
10 else return SolveRec(G, S, Q ∪ {v}, b, x);
```

Algorithm We present a recursive algorithm (see Algorithm 1 for pseudocode) that maintains two sets S and Q. Initially, Algorithm 1 is called with $S = Q = \emptyset$ and we call all pairs of sets S and Q on which a recursive call is made during an execution starting from empty sets as *realizable*. In particular, in such a case, there is an order in which the vertices were added to $S \cup Q$. From now on, we only consider realizable sets S and Q.

For realizable sets S and Q the recursive function is supposed to return a solution to the instance, whenever there is a solution B containing all of S and there is no solution containing S and anything of Q. If some of the conditions is not met, then the function should return "No solution". In other words, S is the set of deleted vertices and Q is the set of vertices the algorithm has decided not to delete in the previous steps but may be collapsed in the future. Hence, the solution to the instance, or the information that there is none, is indeed obtained by calling the recursive function with both sets S and Q empty.

We start by showing that Algorithm 1 has the claimed running time.

Lemma 1 *Algorithm 1 runs in* $O(2^{x+2b}(m+n))$ *time and in* $O(2^{O(b+\sqrt{bx})} \cdot n)$ *time.*

Proof We first show by induction on the size of $S \cup Q$ starting with the largest size achieved that a call with S and Q results in at most $2\binom{2b+x+1-|S|-|Q|}{b-|S|} - 1$ calls to the function in total. Indeed, the size of $|Q|$ never exceeds $b + x + 1$ and the size of $|S|$ never exceeds b since the sizes grow by one at a time and if they achieve the bound, then line 2 or line 5 applies. Hence, if any of the lines 2–6 applies, then we have only one call and $|Q| \leq b+x+1$ and $|S| \leq b$ implies $2\binom{2b+x+1-|S|-|Q|}{b-|S|} - 1 \geq 1$, making the basic cases. If the call makes recursive calls, then in one of them S is one larger than in the current one, and if the second one is made, then Q is one larger in it. Hence

the number of calls is at most $1+2\binom{2b+x+1-|S|-|Q|-1}{b-|S|-1}-1+2\binom{2b+x+1-|S|-|Q|-1}{b-|S|}-1 \leq 2\binom{2b+x+1-|S|-|Q|}{b-|S|} - 1$, finishing the induction.

Since we call the algorithm with sets S and Q empty, it follows that the total number of calls is at most $2\binom{2b+x+1}{b} \leq 2^{2b+x+1} = O(2^{2b+x})$. In particular, if $b = 0$, then the algorithm makes only one call and if $x = 0$ then $2b = 2b+x = 2(b+\sqrt{bx})$. Hence, in both these cases the algorithm makes at most $O(2^{2(b+\sqrt{bx})})$ recursive calls. Otherwise, by a lemma of Fomin et al. [24, Lemma 9] we have that $\binom{2b+x+1}{b} \leq 2^{2\sqrt{b(b+x+1)}}$ which is at most $2^{2(b+\sqrt{bx})}$ as $b(b+x+1) = b^2 + xb + b \leq b^2 + xb + 2b\sqrt{bx} = (b+\sqrt{bx})^2$. So we have that the number of recursive calls is at most $O(2^{2b+x})$ and at most $2^{O(b+\sqrt{bx})}$.

Now we analyze the time complexity of a single call. Lines 2, 4, 5, 6, 7, and 9 can be done in $O(n)$ time. In line 3 the 1-core G' can be found in $O(n+m)$ time by first removing vertices in S and edges incident with them to get $G \setminus S$, and then removing isolated vertices in $G \setminus S$. Thus except for line 8 and 10, all steps can be done in $O(m+n)$ time. Therefore, the algorithm runs in $O(2^{2b+x}(m+n))$ time. Since there are at most $O(bn + x^2)$ edges or we are facing a no-instance, we have that the time complexity of the algorithm is also $O(2^{O(b+\sqrt{bx})} \cdot b^{O(1)}x^{O(1)}n)$, which is $O(2^{O(b+\sqrt{bx})} \cdot n)$. □

Next we show the claimed conditional lower bound on the running time for any algorithm for Collapsed k-Core with $k = 1$.

Lemma 2 *Assuming the Exponential Time Hypothesis, there is no $2^{o(b)}n^{f(x)}$ time algorithm for* Collapsed k-Core *with $k = 1$, for any function f.*

Proof Since Collapsed k-Core with $k = 1$ and $x = 0$ is equivalent to Vertex Cover, and assuming the Exponential Time Hypothesis, there is no $2^{o(k)}n^{O(1)}$ time algorithm for Vertex Cover [34], we have that assuming the Exponential Time Hypothesis, there is no $2^{o(b)}n^{f(x)}$ time algorithm for Collapsed k-Core with $k = 1$, where f can be an arbitrary function. □

Before showing the correctness of Algorithm 1, we first show in the following lemma why in line 2 set Q should be bounded by $b + x$.

Lemma 3 *If S and Q are realizable and Q is of size more than $b + x$, then there is no solution containing the complete set S and no vertex from Q.*

Proof Let S and Q be realizable with $|Q| \geq b + x + 1$. Suppose for contradiction that B is a solution such that $|B| \leq b$, $S \subseteq B$ and $B \cap Q = \emptyset$. Let $v_1, v_2, \ldots, v_{r'}$ be the vertices of the set $S \cup Q$ in the order as they were added to the set by successive recursive calls. If $v_{r'} \in S$, then line 2 would have applied in the parent call and the

current call would never have been executed contradicting realizability of S and Q. Hence $v_{r'} \in Q$. If $B \setminus S$ was empty, then S would be a solution and line 4 would have applied in the parent call, again contradicting realizability of S and Q. Therefore $B \setminus S$ is non-empty and we let $B \setminus S = \{v_{r'+1}, \ldots, v_r\}$, i.e., $\{v_1, \ldots, v_r\} = B \cup Q$. Hence, it always holds that $v_r \in B$. Let G' be the 1-core of $G \setminus B$ and let $X = V(G')$. Since B is a solution, we know that $|X| \leq x$. Our aim is to show that there exists a vertex $v_{j_0} \in Q$ such that after deleting all vertices in $\{v_i \in B \mid i < j_0\}$, deleting vertices in $\{v_i \in B \mid i > j_0\}$ is not enough to make all vertices in $\{v_j \in Q \setminus X \mid j \geq j_0\}$ collapse, which would be a contradiction.

To this end, we construct an injective function f that maps the vertices in B to vertices of $Q \setminus X$. Considering the subsequence containing the vertices of $Q \setminus X$ in the order in which they were added to Q, we assign the last vertex in this sequence to the last vertex of B, the second-to-last to the second-to-last of B, and so on. More formally, for v_r, the last vertex in B, let $f(r)$ be $\max\{j \mid v_j \in Q \setminus X\}$. Now let v_i be the vertex from B with the largest i such that $f(i)$ was not set yet and $v_{i'}$ be the vertex from B with the least i' such that $i' > i$. We set $f(i) = \max\{j \mid j < f(i') \wedge v_j \in Q \setminus X\}$. Since the set $Q \setminus X$ contains at least $b + 1$ vertices, while B contains at most b, this way we find a mapping for every vertex in B. Moreover, there remains at least one vertex in $Q \setminus X$ not being in the image of f. Denote the one with the largest index v_{q_0}.

For $t \in \{1, \ldots, r\}$ let G_t be the 1-core of the graph $G \setminus (B \cap \{v_1, \ldots, v_{t-1}\})$, i.e., for $t \leq r'$, graph G' obtained on line 3 in the call where v_t was added to $S \cup Q$. For $v \in V(G_t)$ let $\deg_t(v)$ be the degree of the vertex v in G_t. By the way we selected v_t we know that $\deg_t(v_t) \geq \deg_t(v_{t'}) \geq \deg_{t'}(v_{t'})$ for every $t < t'$ with $t \leq r'$. Note also that since G_t is a 1-core, we have $\deg_t(v_t) \geq 1$ for all t.

Now we select the special vertex $v_{j_0} \in Q$. If for every vertex $v_i \in B$ we have $f(i) < i$, then let $i_0 = 0$ and $j_0 = q_0$. Otherwise, let i_0 be the largest i such that $f(i) > i$ and $j_0 = f(i_0)$. We consider the graph G_{j_0}. Let $V(G_{j_0}) = B_0 \uplus Q_0 \uplus Y_0 \uplus X$, where $B_0 = \{v_i \in B \mid i > j_0\}$, $Q_0 = \{v_j \in Q \setminus X \mid j \geq j_0\}$, $Y_0 = V(G_{j_0}) \setminus (B_0 \cup Q_0 \cup X)$ is the set of vertices in $V(G_{j_0})$ that will eventually collapse after the deletion of B_0 and not contained in Q_0, and X is the 1-core of $G \setminus B$. Note that in G_{j_0} vertex v_{j_0} in the only vertex of Q_0 which is not contained in the image set of f restricted to B_0. Thus, we have

$$Q_0 = \{v_{j_0}\} \cup \bigcup_{v_i \in B_0} \{v_{f(i)}\}. \tag{1}$$

According to our selection of j_0, for every $v_i \in B_0$ we have $i_0 < j_0 < f(i) < i$. Thus, for each vertex $v_i \in B_0$ we have that

$$\deg_i(v_i) \leq \deg_{f(i)}(v_{f(i)}). \tag{2}$$

Let us count the number, denoted as N_0, of edges of the form $\{v_i, v_j\}$ in G_{j_0} such that $v_i \in B_0$ and $v_j \in Q_0$. Since each edge of G_{j_0} incident on a vertex of Q_0 must have the other endpoint in B_0, we have that

$$N_0 = \sum_{v_j \in Q_0} \deg_{j_0}(v_j) \geq \sum_{v_j \in Q_0} \deg_j(v_j). \tag{3}$$

On the other hand, since edges towards Q_0 are always counted in $\deg_t(v)$, we have that

$$N_0 \leq \sum_{v_i \in B_0} \deg_i(v_i). \tag{4}$$

Combining (3) and (4), we have that

$$\begin{aligned} 0 &\leq \sum_{v_i \in B_0} \deg_i(v_i) - \sum_{v_j \in Q_0} \deg_j(v_j) \\ &\overset{(1)}{=} -\deg_{j_0}(v_{j_0}) + \sum_{v_i \in B_0} (\deg_i(v_i) - \deg_{f(i)}(v_{f(i)})) \\ &\overset{(2)}{\leq} -\deg_{j_0}(v_{j_0}) + \sum_{v_i \in B_0} 0 < 0, \end{aligned}$$

which is a contradiction. □

Now we have all necessary pieces to prove Proposition 2.

Proof of Proposition 2 To show the correctness of the Algorithm 1, we first show that whenever the algorithm outputs a solution, then this solution is indeed correct. Then we show that whenever there exists a solution, the algorithm also finds a solution.

We show correctness of a returned solution by a leaf-to-root induction on the recursion tree. If Algorithm 1 returns S as a solution in line 4, then it is of size at most b since line 2 does not apply and the 1-core of $G \setminus S$ is of size at most x. This constitutes the base case of the induction.

If the solution is obtained from recursive calls on lines 8-10, then we know that it is correct by induction hypothesis.

Next, we show by a leaf-to-root induction on the recursion tree that if there is a solution then Algorithm 1 returns a solution. In particular, we show that if there is a recursive call of Algorithm 1 with realizable sets S and Q such that there is a solution containing all of S and there is no solution containing the whole set S and any vertex from Q, then the algorithm either directly outputs a solution or it invokes a recursive call with sets S' and Q' such that there is a solution containing all of S' and there is no solution containing the whole set S' and any vertex from Q'.

Let S and Q be two realizable input sets of a recursive call of Algorithm 1 and let B be a solution such that $S \subseteq B$ and $B \cap Q = \emptyset$. First we show that none of lines 2, 5, and 6 applies. Line 2 will not apply since we have $|S| \leq |B| \leq b$ and by

Lemma 3 we also know that we have $|Q| \leq b + x$. If G', the 1-core of $G \setminus S$, is of size more than x, which is the case when line 4 does not apply, then $S \neq B$ and line 5 does not apply. If, in this case, line 6 applies, then G' is also the 1-core of $G[Q]$ and no set B' with $B' \cap Q = \emptyset$ can make the 1-core of $G \setminus B'$ of size at most x, which is a contradiction to B being a solution with $B \cap Q = \emptyset$. It follows that the current recursive call of Algorithm 1 does not output "No Solution".

If $B = S$, then line 4 applies and the algorithm outputs B. Otherwise, we know that any vertex v is either in B and hence $S \cup \{v\} \subseteq B$, or we have that $B \cap \{Q \cup \{v\}\} = \emptyset$. In particular, if the recursive call on line 8 does not return a solution, then, by induction hypothesis, there is no solution containing $S \cup \{v\}$, i.e., there is no solution containing S and anything of $Q \cup \{v\}$ and the call on line 10 must return a solution by induction hypothesis. It follows that the algorithm invokes a recursive call with the desired properties in line 8 or in line 10.

Since Algorithm 1 starts with $S = Q = \emptyset$ we have that for any solution B the conditions $S \subseteq B$ and $B \cap Q = \emptyset$ are initially fulfilled. Furthermore, it follows from the complexity analysis in Lemma 1 that the algorithm always terminates. Therefore the algorithm outputs a solution if one exists and hence it is correct.

The running time bound for Algorithm 1 follows from Lemma 1 and the conditional running time lower bound for COLLAPSED k-CORE with $k = 1$ follows from Lemma 2. □

3.3 Algorithm for $k = 2$

Now we show how to adapt the algorithm for $k = 1$ to an algorithm for COLLAPSED k-CORE with $k = 2$.

Theorem 2 COLLAPSED k-CORE *with* $k = 2$ *can be solved in* $O(1.755^{x+4b} \cdot n)$ *time and in* $O(2^{O(b+\sqrt{bx})} \cdot n)$ *time. Assuming the Exponential Time Hypothesis, there is no* $2^{o(b)} n^{f(x)}$ *time algorithm for* COLLAPSED k-CORE *with* $k = 2$, *for any function* f.

The above theorem in particular yields an $O(9.487^b \cdot n)$ algorithm for FEEDBACK VERTEX SET.

Algorithm Our algorithm for $k = 2$ is similar to Algorithm 1 with two main differences (see Algorithm 2 for pseudocode). First $|Q| > b + x$ is replaced by $|Q| > 3b + x$. Second when selecting the maximum degree vertex from $V(G') \setminus Q$, we need to make sure that this vertex has degree greater than 2. Otherwise, either we can directly select vertices from $V(G') \setminus Q$ to break cycles in G' and get a 2-core of size at most x, or the algorithm rejects this branch. We again assume through the section that the function is called with realizable sets S and Q, that is, this call is part of an execution of Algorithm 2 initially called with $S = Q = \emptyset$.

Algorithm 2 Algorithm for COLLAPSED k-CORE with $k = 2$.

```
1  SolveRec2(G, S, Q, b, x)
2  if |S| > b or |Q| > 3b + x then return No solution;
3  Let G' be the 2-core of G \ S.
4  if |V(G')| ≤ x then return S;
5  if |S| = b then return No solution;
6  if V(G') ⊆ Q then return No solution;
7  Let v be a vertex with the highest degree in G' which is not in Q
8  if deg_{G'}(v) ≤ 2 then
9      Let C_1, ..., C_r be the connected components (cycles) of G' not containing
       any vertex of Q, ordered such that |V(C_1)| ≥ |V(C_2)| ≥ ... ≥ |V(C_r)|;
10     Let r' ← min{r, b − |S|};
11     if |V(G')| − Σ_{i=1}^{r'} |V(C_i)| ≤ x then
12         for i = 1, ..., r' do Select an arbitrary vertex from C_i and add it to S;
13         return S
14     else return No solution;
15 T ← SolveRec2(G, S ∪ {v}, Q, b, x) ;
16 if T ≠ No solution then return T;
17 else return SolveRec2(G, S, Q ∪ {v}, b, x) ;
```

We start by claiming that Algorithm 2 has the following running time.

Lemma 4 *Algorithm 2 runs in $O(1.755^{x+4b} \cdot n)$ time and in $O(2^{O(b+\sqrt{bx})} \cdot n)$ time.*

Proof We again first show by induction on the size of $S \cup Q$ starting with the largest size achieved that a call with S and Q results in at most $2\binom{4b+x+1-|S|-|Q|}{b-|S|} - 1$ calls to the function in total. Indeed, the size of $|Q|$ never exceeds $3b+x+1$ and the size of $|S|$ never exceeds b since the sizes grow by one at a time and if they achieve the bound, then line 2 or line 5 applies. Hence, if any of the lines 2–14 applies, then we have only one call and $|Q| \leq 3b+x+1$ and $|S| \leq b$ implies $2\binom{4b+x+1-|S|-|Q|}{b-|S|} - 1 \geq 1$, making the basic cases. If the call makes recursive calls, then in one of them S is one larger than in the current one, and if the second one is made, then Q is one larger in it. Hence the number of calls is at most $1+2\binom{4b+x+1-|S|-|Q|-1}{b-|S|-1} - 1 + 2\binom{4b+x+1-|S|-|Q|-1}{b-|S|} - 1 \leq 2\binom{4b+x+1-|S|-|Q|}{b-|S|} - 1$, finishing the induction.

Since we again call the algorithm with sets S and Q empty, it follows that the total number of calls is at most $2\binom{4b+x+1}{b}$. Using, e.g., the lemma of Fomin et al. [24, Lemma 10] (or Stirling's approximation) one can show that $\binom{z}{\frac{1}{4}z} = O(1.7549^z)$. Hence, $\binom{4b+x+1}{b} \leq \binom{4b+x+1}{\frac{1}{4}(4b+x+1)} = O(1.7549^{4b+x+1})$.

If $b = 0$, then the algorithm makes only one call and if $x = 0$ then $4b + 1 = 4b + x + 1 = O(b + \sqrt{bx})$. Hence, in both these cases the algorithm makes at most $2^{O(b+\sqrt{bx})}$ recursive calls. Otherwise, again by the other lemma of Fomin et al. [24, Lemma 9] we have that $\binom{4b+x+1}{b} \leq 2^{2\sqrt{b(3b+x+1)}}$. which is at most $2^{2\sqrt{3}\cdot(b+\sqrt{bx})}$ as

$b(3b+x+1) = 3b^2+xb+b \leq 3b^2+xb+6b\sqrt{bx}+2xb = 3(b+\sqrt{bx})^2$. So we have that the number of recursive calls is at most $O(1.7549^{4b+x})$ and at most $2^{O(b+\sqrt{bx})}$.

Now we analyze the time complexity of a single call. In line 3, the 2-core G' can again be found in $O(m)$ time by recursively removing vertices with degree less than 2 in $G \setminus S$ [2]. In line 9, sorting these cycles according to their sizes can be done in $O(n)$ time using counting sort. Thus except for line 15 and 17, all steps can again be done in $O(m+n)$ time. Since there are at most $O(bn+x^2)$ edges or we are facing a no-instance, we have that the time complexity of the algorithm is $O(1.7549^{4b+x} \cdot b^{O(1)}x^{O(1)}n)$ and $O(2^{O(b+\sqrt{bx})} \cdot b^{O(1)}x^{O(1)}n)$, which is $O(1.755^{4b+x} \cdot n)$ and $O(2^{O(b+\sqrt{bx})} \cdot n)$ as claimed. □

Next we claim the following conditional lower bound on the running time for any algorithm for COLLAPSED k-CORE with $k = 2$.

Lemma 5 *Assuming the Exponential Time Hypothesis, there is no $2^{o(b)}n^{f(x)}$ time algorithm for* COLLAPSED k-CORE *with $k = 2$, for any function f.*

Proof Since COLLAPSED k-CORE with $k = 2$ and $x = 0$ is equivalent to FEEDBACK VERTEX SET, and assuming the Exponential Time Hypothesis, there is no $2^{o(k)}n^{O(1)}$ time algorithm for FEEDBACK VERTEX SET [34], we have that assuming the Exponential Time Hypothesis, there is no $2^{o(b)}n^{f(x)}$ time algorithm for COLLAPSED k-CORE with $k = 2$, where f is an arbitrary function. □

Before showing the correctness of Algorithm 2, we give the following lemmata which will be helpful in the correctness proof. The following lemma claims that, except for some specific connected components, we can limit the solution to contain vertices of degree at least three.

Lemma 6 *For any instance $(G, b, x, 2)$ of* COLLAPSED k-CORE *with $k = 2$, where G is a 2-core and does not contain a cycle as a connected component, if there is a solution B for $(G, b, x, 2)$, then there is also a solution B' for $(G, b, x, 2)$ which contains only vertices with degree larger than 2.*

Proof Let v be any vertex with degree 2 in B. Let v' be a vertex of degree at least 3 in G such that there is a path P between v and v' with all internal vertices of degree exactly 2 in G. Since no component of G is a cycle, such a vertex must exist. Let $B_1 = B \cup \{v'\}$ and $B_2 = (B \cup \{v'\}) \setminus \{v\}$. We have that the 2-core of $G \setminus B_2$ is the same as the 2-core of $G \setminus B_1$ and the 2-core of $G \setminus B_1$ is a subset of the 2-core of $G \setminus B$. So the 2-core of $G \setminus B_2$ is a subset of the 2-core of $G \setminus B$, and hence no larger than x. Therefore B_2 is also a solution for $(G, b, x, 2)$. Following the same way, we can replace all degree 2 vertices in B with vertices which have degree larger than 2, and get a new solution B' for $(G, b, x, 2)$. □

The next lemma helps to show that line 14 is correct.

Lemma 7 *If S and Q are realizable and line 14 of Algorithm 2 applies, and for every $q \in Q$ there is no solution containing the complete set $S \cup \{q\}$, then there is no solution completely containing S.*

Proof Suppose for contradiction that B is a solution containing S. Note that in this case $B' = B \setminus S$ is a solution for $(G', b - |S|, x, 2)$. Let D be the graph formed by the union of connected components of G' containing at least one vertex of degree at least 3 and C_0 be the graph formed by the union of connected components of G' which are cycles and contain vertices of Q, that is, $V(D) \cup V(C_0) = V(G') \setminus \bigcup_{i=1}^{r} V(C_i)$, where C_i is as stated in the algorithm.

If $B^D = B \cap V(D)$ is nonempty and x' is the size of the 2-core of $D \setminus B^D$, then B^D is a solution to the instance $(D, |B^D|, x', 2)$. Hence, by Lemma 6, there is another solution B_3^D to the instance $(D, |B^D|, x', 2)$ which contains only vertices of degree at least 3. However, according to line 8 of Algorithm 2, all vertices of degree at least 3 are in Q, and hence $(B \setminus B^D) \cup B_3^D$ is a solution to $(G, b, x, 2)$ containing the complete set S and vertices of Q, contradicting our assumption. Hence B^D is empty.

If $B^C = B \cap V(C_0)$ is nonempty, then let y be any vertex in B^C and C_y the connected component of C_0 containing y. By the definition of C_0 component C_y contains a vertex of Q, let us denote it y'. Let $\widehat{B} = (B \setminus \{y\}) \cup \{y'\}$. The 2-cores of $G \setminus B$ and $G \setminus \widehat{B}$ are the same, since C_y is a cycle. Thus $\widehat{B}$ is a solution to $(G, b, x, 2)$ containing the complete set S and vertices of Q, contradicting our assumption. Hence also B^C is empty.

The components of G' neither in C_0 nor in D are cycles since G' is a 2-core, they contain no vertices of Q, and there are no other vertices of degree at least 3. Since B contains at most $r' = \min\{r, b - |S|\}$ vertices out of these components, it can destroy at most r' of these cycles. Since $|V(C_1)| \geq |V(C_2)| \geq \ldots \geq |V(C_r)|$, this decreases the size of the 2-core by at most $\sum_{i=1}^{r'} |V(C_i)|$. Thus, if $|V(G')| - \sum_{i=1}^{r'} |V(C_i)| > x$, then there is no solution containing the complete set S. □

Next, we show that the second part of line 2 of Algorithm 2 is correct.

Lemma 8 *If S and Q are realizable and Q is of size more than $3b + x$, then there is no solution containing the complete set S and no vertex from Q.*

Proof Let S and Q be realizable with $|Q| \geq 3b + x + 1$. Suppose for contradiction that B is a solution such that $|B| \leq b$, $S \subseteq B$ and $B \cap Q = \emptyset$. Let $v_1, v_2, \ldots, v_{r'}$ be the vertices of the set $S \cup Q$ in the order as they were added to the set by successive recursive calls. Similarly to the situation when $k = 1$, it always holds that $B \setminus S$ is non-empty. We let $B \setminus S = \{v_{r'+1}, \ldots, v_r\}$, i.e., $\{v_1, \ldots, v_r\} = B \cup Q$. In particular $v_r \in B$. Without loss of generality, we can assume that for every vertex $v_i \in B$, v_i is contained in the 2-core of $G \setminus (B \cap \{v_1, v_2, \ldots, v_{i-1}\})$. For $i \leq r'$, this is clear since v_i is selected from G' in line 7. For $i > r'$, $v_i \in B \setminus S$, and if v_i is not contained in the 2-core of $G \setminus (B \cap \{v_1, v_2, \ldots, v_{i-1}\})$, then $B' = B \setminus \{v_i\}$ is also a solution such that $|B'| \leq b$, $S \subseteq B'$ and $B' \cap Q = \emptyset$. Let G' be the 2-core of $G \setminus B$ and let $X = V(G')$. Since B is a solution, we know that $|X| \leq x$. Our

aim is to show that there exists a vertex $v_{j_0} \in Q$ such that after deleting all vertices in $\{v_i \in B \mid i < j_0\}$, deleting vertices in $\{v_i \in B \mid i > j_0\}$ is not enough to make all vertices in $\{v_j \in Q \setminus X \mid j \geq j_0\}$ collapse, which would be a contradiction.

To this end, we construct a function f that maps every vertex of B to a set of three consecutive vertices of $Q \setminus X$ (see also Fig. 5). Considering the subsequence containing the vertices of $Q \setminus X$ in the order in which they were added to Q, we split the last $3b$ vertices of this sequence into consecutive triples, and assign the last triple to the last vertex of B, the second-to-last to the second-to-last of B, and so on. More formally, for v_r, the last vertex in B, let $f(r)$ be the set $\{j_1, j_2, j_3\}$, where $j_1 = \max\{j \mid v_j \in Q \setminus X\}$, $j_2 = \max\{j < j_1 \mid v_j \in Q \setminus X\}$, and $j_3 = \max\{j < j_2 \mid v_j \in Q \setminus X\}$. Now let v_i be the vertex from B with the largest i such that $f(i)$ was not set yet and $v_{i'}$ be the vertex from B with the least i' such that $i' > i$. We set $f(i) = \{j_1, j_2, j_3\}$, where $j_1 = \max\{j < \min\{k \in f(i')\} \mid v_j \in Q \setminus X\}$, $j_2 = \max\{j < j_1 \mid v_j \in Q \setminus X\}$, and $j_3 = \max\{j < j_2 \mid v_j \in Q \setminus X\}$. Since the set $Q \setminus X$ contains at least $3b + 1$ vertices, while B contains at most b, this way we find a mapping for every vertex in B, keeping the images of different vertices disjoint. Moreover, denote $p = |Q \setminus X| - 3|B| \geq 1$. There remain p vertices in $Q \setminus X$ not being in the union of images of f, let us denote them $v_{q_1}, \ldots, v_{q_p}$.

For $t \in \{1, \ldots, r\}$ let G_t be the 2-core of the graph $G \setminus (B \cap \{v_1, \ldots, v_{t-1}\})$, i.e., for $t \leq r'$, graph G' obtained on line 3 in the call where v_t was added to $S \cup Q$. For $v \in V(G_t)$ let $\deg_t(v)$ be the degree of the vertex v in G_t. By the way we selected v_t we again know that $\deg_t(v_t) \geq \deg_t(v_{t'}) \geq \deg_{t'}(v_{t'})$ for every $t < t'$ with $t \leq r'$. Note also that $\deg_t(v_t) \geq 3$ for all $v_t \in Q$.

Now we select the special vertex $v_{j_0} \in Q$. If for every vertex $v_i \in B$ we have $i > \max\{j \mid j \in f(i)\}$, then let $i_0 = 0$ and $j_0 = q_p$. Otherwise, let i_0 be the largest i such that $i < \max\{j \mid j \in f(i)\}$ and $j_0 = \max\{j \mid j \in f(i_0)\}$. We consider the graph G_{j_0} (see also Fig. 5). Let $V(G_{j_0}) = B_0 \uplus Q_0 \uplus Y_0 \uplus X$, where $B_0 = \{v_i \in B \mid i > j_0\}$, $Q_0 = \{v_j \in Q \setminus X \mid j \geq j_0\}$, $Y_0 = V(G_{j_0}) \setminus (B_0 \cup Q_0 \cup X)$ is the set of vertices in $V(G_{j_0})$ that will eventually collapse after the deletion of B_0 and not contained in Q_0, and X is the 2-core of $G \setminus B$. Note that in G_{j_0} vertex v_{j_0} is the only vertex of Q_0 which is not contained in any image set of f restricted to B_0. Thus, we have

$$Q_0 = \{v_{j_0}\} \cup \bigcup_{v_i \in B_0} \bigcup_{j \in f(i)} \{v_j\}. \tag{5}$$

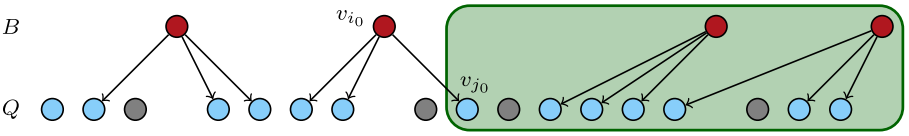

Fig. 5 Illustration of function f. Vertices in Q are separated into two parts: gray vertices from X and blue vertices from $Q \setminus X$. Every vertex in B is mapped to a set of three consecutive blue vertices in $Q \setminus X$ from the right to the left. Graph G_{j_0} with the property that $i > \max\{j \mid j \in f(i)\}$ for every i with $i > j_0$ is contained in the green box

According to our selection of j_0, for every i with $v_i \in B_0$ we have $i > \max\{j \mid j \in f(i)\}$. Hence, $\deg_i(v_i) \leq \deg_j(v_j)$ for every vertex $v_i \in B_0$ and every $j \in f(i)$. Together with $\deg_t(v_t) \geq 3$ for all $v_t \in Q$, we have

$$\deg_i(v_i) - \sum_{j \in f(i)} \deg_j(v_j) + 6 \leq 0 \tag{6}$$

for all $v_i \in B_0$.

Now we transform graph G_{j_0} into a partial directed graph by considering the collapsing process (see also Fig. 6). More precisely, for every edge in G_{j_0}, except for edges which have both endpoints in X, we will assign it a direction. To this end, we just need to give an order of vertices in $V(G_{j_0}) \setminus X$. Then we direct each edge from the vertex appearing earlier in the order to the one that appears later. We define this order based on the time vertices are being deleted (i.e., respecting the order $v_1, v_2, \ldots, v_r$ on B_0) or collapsed. It may happen that several vertices in Q_0 or Y_0 collapse at the same time. For this situation, we just order these vertices according to an arbitrary but fixed order. Since $k = 2$, every collapsed vertex in $Q_0 \cup Y_0$ has at most one outgoing edge.

Then we consider the number, denoted by N_0, of edges of G_{j_0} of the form $\overrightarrow{v_i v_j}$ such that the head v_j is in Q_0. Since every vertex in Q_0 has at most one outgoing edge and $\deg_{j_0}(v_j) \geq \deg_j(v_j)$ for all $v_j \in Q_0$, we have

$$N_0 \geq \sum_{v_j \in Q_0} (\deg_{j_0}(v_j) - 1) \geq \sum_{v_j \in Q_0} \deg_j(v_j) - |Q_0|. \tag{7}$$

Now for any two sets $M_1, M_2 \subseteq V(G_{j_0})$, let $N_{\overrightarrow{M_1 M_2}}$ be the number of edges going from M_1 to M_2. Denote $\deg^+(v)$ and $\deg^-(v)$ the number of incoming and outgoing edges of vertex v in G_{j_0}, respectively. We first claim that

$$N_{\overrightarrow{Y_0 Q_0}} \leq N_{\overrightarrow{B_0 Y_0}} + N_{\overrightarrow{Q_0 Y_0}}. \tag{8}$$

To show that, note that every vertex in Y_0 has at least one incoming edge but at most one outgoing edge, implying that $\sum_{v \in Y_0} \deg^-(v) \leq \sum_{v \in Y_0} \deg^+(v)$. This means that

$$N_{\overrightarrow{Y_0 B_0}} + N_{\overrightarrow{Y_0 Q_0}} + N_{\overrightarrow{Y_0 Y_0}} + N_{\overrightarrow{Y_0 X}} \leq N_{\overrightarrow{B_0 Y_0}} + N_{\overrightarrow{Q_0 Y_0}} + N_{\overrightarrow{Y_0 Y_0}}.$$

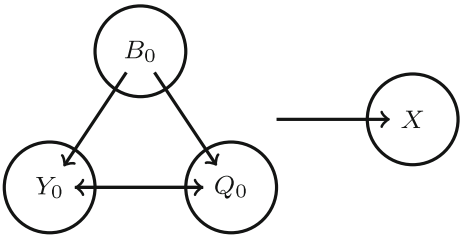

Fig. 6 Illustration of the partial directed graph when considering the collapsing process. The set B_0 contains the deleted vertices and X is the set of vertices remaining in the 2-core of $G \setminus B$. The set Q_0 contains vertices the algorithm has decided not to delete but eventually collapse and Y_0 is the set of other eventually collapsed vertices

Therefore, $N_{\overrightarrow{Y_0Q_0}} \leq N_{\overrightarrow{B_0Y_0}} + N_{\overrightarrow{Q_0Y_0}}$, as claimed.

Now we give an upper bound for N_0. Since the edges which have their heads in Q_0 have their tails from $B_0 \cup Y_0 \cup Q_0$, we have

$$\begin{aligned} N_0 &= N_{\overrightarrow{B_0Q_0}} + N_{\overrightarrow{Y_0Q_0}} + N_{\overrightarrow{Q_0Q_0}} \\ &\leq N_{\overrightarrow{B_0Q_0}} + N_{\overrightarrow{Y_0Q_0}} + (|Q_0| - N_{\overrightarrow{Q_0Y_0}}) \\ &\overset{(8)}{\leq} N_{\overrightarrow{B_0Q_0}} + (N_{\overrightarrow{B_0Y_0}} + N_{\overrightarrow{Q_0Y_0}}) + (|Q_0| - N_{\overrightarrow{Q_0Y_0}}) \\ &= N_{\overrightarrow{B_0Q_0}} + N_{\overrightarrow{B_0Y_0}} + |Q_0| \\ &\leq \sum_{v_i \in B_0} \deg^-(v_i) + |Q_0| \\ &\leq \sum_{v_i \in B_0} \deg_i(v_i) + |Q_0|. \end{aligned} \tag{9}$$

The last inequality holds since for every vertex $v_i \in B_0$, v_i is contained in the 2-core of $G_{j_0} \setminus (B_0 \cap \{v_1, v_2, \ldots, v_{i-1}\})$, which means all outgoing edges of v_i are counted in $\deg_i(v_i)$. Thus we have that $\sum_{v_i \in B_0} \deg^-(v_i) \leq \sum_{v_i \in B_0} \deg_i(v_i)$. Therefore, putting (7), (9), and finally (6) together we have

$$\begin{aligned} 0 &= N_0 - N_0 \\ &\leq \sum_{v_i \in B_0} \deg_i(v_i) + |Q_0| - \left(\sum_{v_j \in Q_0} \deg_j(v_j) - |Q_0| \right) \\ &= \sum_{v_i \in B_0} \deg_i(v_i) - \sum_{v_j \in Q_0} \deg_j(v_j) + 2|Q_0| \\ &= \sum_{v_i \in B_0} \deg_i(v_i) - \sum_{v_j \in Q_0} (\deg_j(v_j) - 2) \\ &\overset{(5)}{=} \sum_{v_i \in B_0} \deg_i(v_i) - \left(\sum_{v_i \in B_0} \sum_{j \in f(i)} (\deg_j(v_j) - 2) + \deg_{j_0}(v_{j_0}) - 2 \right) \\ &\overset{|f(i)|=3}{=} \sum_{v_i \in B_0} \left(\deg_i(v_i) - \sum_{j \in f(i)} \deg_j(v_j) + 6 \right) - \deg_{j_0}(v_{j_0}) + 2 \\ &\overset{(6)}{\leq} 0 - \deg_{j_0}(v_{j_0}) + 2 \\ &< 0, \end{aligned}$$

which is a contradiction. The last inequality holds since $\deg_t(v_t) \geq 3$ for all $v_t \in Q$. □

Proof of Theorem 2 To show the correctness of the algorithm, it is again enough to show that for any realizable pair of S and Q appearing in the recursive process, the set returned by the recursive function is a solution and if there is a solution containing the complete set S and there is no solution containing the whole set S and any

vertex from Q, then the function returns a solution. We do this again by induction starting from the leaves of the recursion tree. For simplicity we assume without loss of generality that the input graph is a 2-core.

$\Leftarrow$: If the function returns S as a solution on line 4, then it is of size at most b since line 2 does not apply and the 2-core of $G \setminus S$ is of size at most x. Hence it is obviously a solution and it is fulfilling the constraints. If the solution is obtained from recursive calls on lines 15-17, then it is a solution by induction hypothesis.

If the function returns set S' as a solution on line 13, then for each $i \in \{1, \ldots, r'\}$ set S' contains a vertex of the cycle C_i. Hence the 2-core of $G \setminus S'$ does not contain the cycle C_i. Since the 2-core G'' of $G \setminus S'$ differs from the 2-core G' of $G \setminus S$ exactly in these missing cycle components, we have $V(G'') = |V(G')| - \sum_{i=1}^{r'} |V(C_i)|$. Since this is at most x by line 11 and line 2 does not apply, S is a solution. Hence, if the function returns a solution, then the answer is correct.

$\Rightarrow$: Now we show by induction on $|Q \cup S|$ for every realizable pair of S and Q, starting from the largest $|Q \cup S|$ achieved, that if there is a solution containing the complete set S and there is no solution containing the whole set S and any vertex from Q, then the algorithm returns a solution.

If B is a solution such that $S \subseteq B$ and $B \cap Q = \emptyset$, then line 2 will not apply since $|B| \leq b$ and because of Lemma 8, whereas line 14 does not apply according to Lemma 7.

Let G' be the 2-core of $G \setminus S$. If B is a solution, then $S \cup ((B \setminus S) \cap V(G'))$ is also a solution, so we can assume that $B \setminus (S \cup V(G')) = \emptyset$. If $B = S$, then line 4 applies. Otherwise, line 5 does not apply. If there are no vertices of degree at least 3 which are not in Q, then B contains no vertices of the components containing vertices of Q, as we have shown in the proof of Lemma 7 that B^D is empty. Hence B can decrease the size of the 2-core compared to G' by at most $\sum_{i=1}^{r'} |V(C_i)|$, where r' is as in the algorithm. As B is a solution, this implies $|V(G')| - \sum_{i=1}^{r'} |V(C_i)| \leq x$ and a solution containing S is returned on line 13. This finishes the proof for the base cases of the induction.

Now suppose that the claim already holds for all calls with larger $|Q \cup S|$ and v is the vertex selected by the algorithm on line 7. If there is a solution containing $S \cup \{v\}$ (in particular if $v \in B$), then the call SolveRec2(G, $S \cup \{v\}$, Q) must return a solution by induction hypothesis and otherwise there is no solution containing S and anything of $Q \cup \{v\}$ and the call SolveRec2(G, S, $Q \cup \{v\}$) must return a solution by induction hypothesis. Thus the algorithm works correctly.

The running time bound for Algorithm 2 follows from Lemma 4 and the conditional running time lower bound for Collapsed k-Core with $k = 2$ follows from Lemma 5. □

3.4 Complexity on Cubic Graphs

Feedback Vertex Set is polynomial time solvable on graphs of maximum degree 3 [10, 43]. In fact, this is true even if some vertices of the graph cannot be taken into a solution and the bound on the maximum degree only holds for vertices that can take part in the solution [10]. This allows Cao [9] to improve his algorithm, which is very similar to Algorithm 2, by directly solving the instance already

if the degree of all vertices of G' not in Q is at most 3, achieving a running time of $O(8^b \cdot n^{O(1)})$.[5] If also COLLAPSED k-CORE with $k = 2$ could be solved in polynomial time on graphs, where all vertices (that can be taken into solution) have degree at most 3, then this would improve also the running time of our algorithm for COLLAPSED k-CORE with $k = 2$ on general graphs.

In this subsection we show that the result of Cao et al. [10] probably does not generalize to COLLAPSED k-CORE with $k = 2$, at least not in its full generality. To this end, we need the following generalization of our problem:

COLLAPSED 2-CORE WITH FORBIDDEN VERTICES

Input: An undirected graph $G = (V, E)$, a partition of V into V_1 and V_2, and integers b and x.

Question: Is there a set $S \subseteq V_1$ with $|S| \leq b$ such that the 2-core of $G - S$ has size at most x?

Theorem 3 COLLAPSED 2-CORE WITH FORBIDDEN VERTICES *is NP-hard, even on graphs of maximum degree 3, even if the degree of all vertices in V_1 is 2.*

An attentive reader might notice that Algorithm 2 actually deals with instances similar to those produced by Theorem 3 in polynomial time. We postpone the discussion of how this is possible after the proof of the theorem.

Proof of Theorem 3 We provide a polynomial time reduction from CLIQUE in regular graphs, which is NP-hard [39]. Let (G, s) be an instance of CLIQUE such that G is r-regular for some r. We assume that $s \geq 1$ and, hence, $n - s + 1 \leq n$, as otherwise the instance is trivial. We construct an instance (G', V_1, V_2, b, x) of COLLAPSED 2-CORE WITH FORBIDDEN VERTICES as follows. Assume without loss of generality that $V(G) = \{v_1, \ldots, v_n\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$.

We start with G' being a simple cycle on vertices $C = \{c_1, \ldots, c_m\}$. Then we introduce to G', for every $j \in \{1, \ldots m\}$, a vertex d_j and connect it by an edge to c_j. We let $D = \{d_1, \ldots, d_m\}$. For every $j \in \{1, \ldots m\}$ let us denote the endpoints of edge e_j as v_x and v_y. We introduce two vertices $f_{x,y}$ and $f_{y,x}$ and let both of them be adjacent to d_j. For every $i \in \{1, \ldots, n\}$ we let $F_i = \{f_{i,y} \mid y \in \{1, \ldots, n\}, \{v_i, v_y\} \in E(G)\}$. We introduce a rooted binary tree T_i with leaves in F_i, root in a new vertex h_i, and all internal vertices (except for h_i) having degree 3. All the internal vertices are newly introduced to G'. Finally, we introduce a simple cycle on vertices $A = \{a_1, \ldots, a_n\}$ and connect for each $i \in \{1, \ldots, n\}$ the vertices h_i and a_i by a path P_i with $L - r + 1$ internal vertices, where $L = 5m$. The construction is illustrated on Fig. 7. Let us denote $F = \bigcup_{i=1}^{n} F_i$.

To finish the construction, we let $V_1 = F$, $V_2 = V(G') \setminus F$, $b = rs$, and $x = N - b - x'$, where $x' = sL + \binom{s}{2}$, and $N = V(G')$ is the total number of vertices in the newly constructed graph.

Before we show the correctness of the reduction, let us count the number of vertices in each of the sets. There are m vertices in C and m vertices in D. For each i

[5] Actually, using an analysis similar to that of Lemma 4, one can show that the improved algorithm of Cao runs in $O(6.75^b \cdot n^{O(1)})$ time.

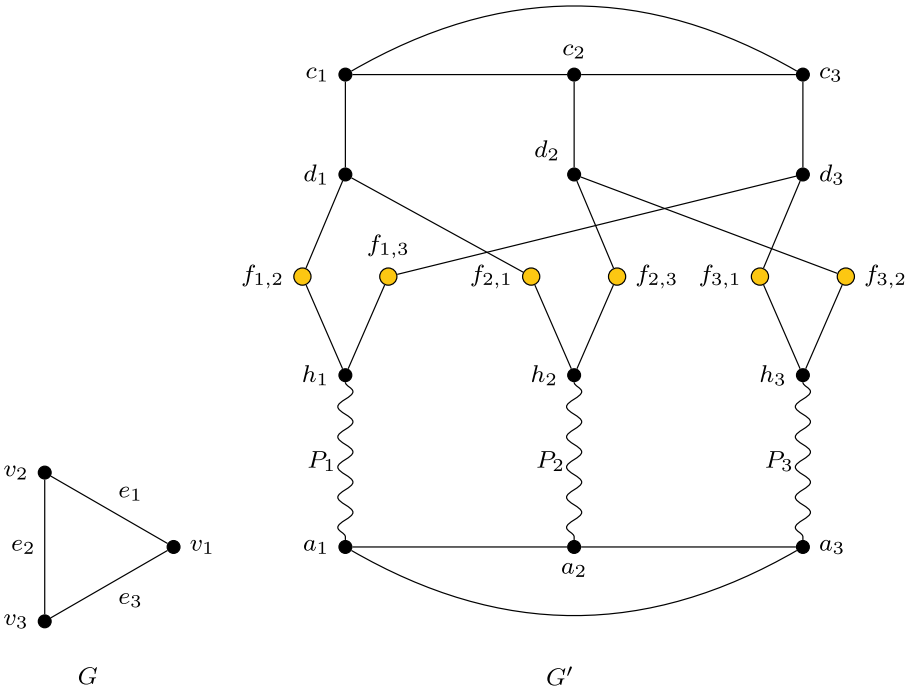

Fig. 7 Illustration of the construction in the proof of Theorem 3 for a 2-regular graph G depicted on the left. The paths P_i in G' (depicted on the right) have $5m - r + 1 = 15 - 2 + 1 = 14$ internal vertices each. For $s = 3$ we would have $b = 2 \cdot 3 = 6$ and $x = N - x' = 12$, as $N = 60$ and $x' = 48$. Yellow vertices of G' are contained in V_1, all other vertices of G' are contained in V_2

there are r vertices in F_i (as each vertex of G is incident with r edges) and, hence, there are exactly $n \cdot r = 2m$ vertices in F in total, by the Handshaking Lemma. As T_i is a rooted binary tree with r leaves, it has $r - 1$ internal vertices (including the root). There are $L - r + 1$ internal vertices on each P_i. Therefore, for each i, there are $r - 1 + (L - r + 1) = L$ internal vertices together in T_i and P_i. Finally, there are n vertices in A. Hence, $N = m + m + 2m + nL + n = 4m + n + nL$ and, thus, $x = 4m + n - b - \binom{s}{2} + (n - s)L$.

We next show that (G', V_1, V_2, b, x) is a yes-instance of Collapsed 2-Core with Forbidden Vertices if and only if (G, s) is a yes-instance of Clique.

$\Rightarrow$: We start with the "if" direction. Suppose that (G, s) is a yes-instance of Clique and let S be a clique of size s in G. We let $B = \bigcup_{v_i \in S} F_i$. As each F_i is of size exactly r, set B is of size exactly rs. Now let G'' be the 2-core of $G' \setminus B$ and let us count the number of vertices which are neither in B nor in $V(G'')$. For any i such that $v_i \in S$, as the leaves of T_i are in B and all the internal vertices (except for the root) only have neighbors inside the tree, none of them remains in G''. Thus, also the vertex h_i would only have one neighbor in G'' and, hence, neither it, nor the internal vertices on path P_i are a part of G''. This makes $(r - 2) + 1 + (L - r + 1) = L$ vertices which are neither in B nor in $V(G'')$ for each i such that $v_i \in S$.

Furthermore, for each j such that $e_j \in E(G[S])$ the vertex d_j would have at most one neighbor in G'' and thus it is not part of G''. As S is a clique in G, there are $\binom{s}{2}$ edges in $E(G[S])$ and, thus, the total number of vertices from D which are not in G'' is $\binom{s}{2}$. Therefore, in total, there are at least $s \cdot L + \binom{s}{2} = x'$ vertices which are neither in B, nor in G''. Hence, $|V(G'')| \leq N - b - x' = x$ and B is a solution to (G', V_1, V_2, b, x).

$\Leftarrow$: Now for the "only if" direction let us assume that there is a solution B to (G', V_1, V_2, b, x) and let G'' be the 2-core of $G' \setminus B$. First notice that the cycles A and C are always contained in G'' as $B \subseteq F$. Let us now take some $i \in \{1, \ldots, n\}$ such that $F_i \setminus B \neq \emptyset$ and let $f_{i,y} \in F_i \setminus B$. Moreover, let $e_j = \{v_i, v_y\}$. The edge between c_j and d_j, the edge connecting d_j and $f_{i,y}$, the path in T_i from $f_{i,y}$ to h_i, and the path P_i form together a path connecting the cycles A and C in $G' \setminus B$ and, hence, also in G''. Note that the part of this path between $f_{i,y}$ and a_i, including $f_{i,y}$ and h_i, but not including a_i, has at least $(L - r + 1) + 2 = L - r + 3$ vertices and cannot be shared between different indices i. Let $S = \{v_i \mid F_i \setminus B = \emptyset\}$. If $|S| < s$, then there are at least $n - s + 1$ indices i such that $F_i \setminus B \neq \emptyset$. That is,

$$\begin{aligned} |V(G'')| &\geq m + n + (n - s + 1)(L - r + 3) \\ &> m + n + (n - s)L + L - (n - s + 1)r \\ &\geq 4m + n - b - \binom{s}{2} + (n - s)L \\ &= x \end{aligned}$$

since

$$L - (n - s + 1)r \geq L - nr = 5m - 2m > 3m - b - \binom{s}{2}.$$

Hence, $|S| \geq s$, and, as $rs \geq |B| \geq r|S|$ by the definition of S, we have $|S| = s$. That is, $B = \bigcup_{v_i \in S} F_i$.

For every i such that $v_i \notin S$, G'' contains F_i, T_i, and P_i by the above argument. This makes $r + r - 1 + (L - r + 1) = L + r$ vertices for each such i (not counting a_i) and $(n - s)(L + r) = nr - rs + (n - s)L = 2m - b + (n - s)L$ vertices together. Moreover, the cycles on A and C are always contained in G''. If there are more than $m - \binom{s}{2}$ indices j such that d_j is in G'', then

$$\begin{aligned} |V(G'')| &> m + n + 2m - b + (n - s)L + \left(m - \binom{s}{2}\right) \\ &= 4m + n - b - \binom{s}{2} + (n - s)L \\ &= x. \end{aligned}$$

This would contradict B being a solution.

Hence, there are at least $\binom{s}{2}$ indices j such that d_j is not in G''. For each such j, letting $e_j = \{v_x, v_y\}$, we must have $f_{x,y} \in B$ and also $f_{y,x} \in B$, as otherwise d_j would have at least 2 neighbors in $G'' - c_j$ and at least one of $f_{y,x}$ and $f_{x,y}$. Hence, both $v_x \in S$ and $v_y \in S$, i.e., $e_j \in E(G[S])$. It follows that $|E(G[S])| \geq \binom{s}{2}$, that is S is a clique of size s in G and (G, s) is a yes-instance of CLIQUE.

Since the reduction can be performed in polynomial time, all vertices have degree at most 3 in G', vertices in $V_1 = F$ have degree 2, and the instances are equivalent, this finishes the proof. □

An instance similar to the one constructed in Theorem 3 might actually occur during the run of Algorithm 2, i.e., G' might be the 2-core of $G \setminus S$ produced in step 3 of the algorithm with $Q = V_2 = V(G') \setminus F$. In particular, Algorithm 2 imposes no relation between $|Q|$ and b, as the relation is actually between $|Q|$ and $|S| + b$. The reason Algorithm 2 is able to deal with the instance in polynomial time is that we require it to solve the instance only if there is no solution containing any of the vertices of $Q = V_2$. Hence, it does not solve the instance, but merely rejects it, or, more precisely, does not search for vertices of the solution in this component (see Lemma 6).

The current algorithm deals with instances where all vertices of degree more than 2 in the 2-core belong to Q, in polynomial time. To generalize this in the most direct way, allowing the algorithm to deal in polynomial time with cases where all vertices of degree more than 3 in the 2-core belong to Q, we would need at least two ingredients:

(i) If there is a solution to the component and there are vertices of degree at least 4 in the component, then there is a solution containing some vertex of degree at least 4 (an analogue of Lemma 6).
(ii) A solution for each component of maximum degree at most 3, not containing any vertices of Q, can be found in polynomial time (an analogue of steps 8–14 of Algorithm 2).

Concerning (i), consider graph obtained from a disjoint union of a cycle C_n and K_4 by adding an edge between an arbitrary vertex of the cycle and an arbitrary vertex of the K_4. The resulting graph has exactly one vertex of degree 4 (or more). However, for $n \geq 5$, $b = 1$, and $x = 4$ this vertex is not part of any solution. Indeed, each solution is formed by a vertex of the cycle C_n.

Concerning (ii), note specifically that it is not enough to find a minimum feedback vertex set of the component, as it may contain too many vertices. In particular, considering graph G' constructed in Theorem 3 (without forbidden vertices), removing n vertices of A is sufficient to reduce the 2-core of the graph from $\Omega(nm)$ to $O(m)$ vertices, whereas the minimum feedback vertex set is of size $\Theta(m)$. The budget spared on this component might help reduce the 2-core more significantly on some other component, which might have similar properties. This was never an issue with components of maximum degree 2, as the minimum feedback vertex set was always of size 1. Nevertheless, we leave open whether (ii) is true.

Hence, significantly new ideas will be needed to improve the running time of Algorithm 2.

4 Structural Graph Parameters

In this section, we investigate the parameterized complexity of COLLAPSED k-CORE with respect to several structural parameters of the input graph. Corollary 1 already

implies hardness for constant values of several structural graph parameters. We expand this picture by observing that the problem remains NP-hard on graphs with a dominating set of size one and by showing that the problem is W[1]-hard when parameterized by the combination of b and the clique cover number of the input graph. On the positive side, we show that the problem is in FPT when parameterized by the treewidth of the input graph or the clique-width of the input graph and k combined with either b, x, $n - x$, or $n - b$. Lastly, we show that the problem presumably does not admit a polynomial kernel for any $k \geq 2$ when parameterized by the combination of b and the vertex cover number of the input graph, or when parameterized by the combination of b, $n - x$, k, and the bandwidth of the input graph.

We start with an easy observation that we will make use of in most of the hardness results in this section.

Observation 2 *If (G, b, x, k) is an instance of* Collapsed k-Core *and vertex v is a part of the $(k + b)$-core of G, and $S \subseteq V$ is of size at most b, then either $v \in S$ or v is part of the k-core of $G \setminus S$.*

Proof Let C be the $(k + b)$-core of G. In $C \setminus S$ the degree of each vertex is at least $k + b - b$, hence $C \setminus S$ is a subgraph of the k-core of $G \setminus S$. □

The following observation yields that we can reduce the size of a dominating set of any instance of Collapsed k-Core to one by introducing a universal vertex. Note that, for example, this only increases the degeneracy by one.

Observation 3 *Let (G, b, x, k) be an instance of* Collapsed k-Core *and G' be the graph obtained from G by adding a universal vertex, then $(G', b + 1, x, k)$ is an equivalent instance of* Collapsed k-Core.

Proof Let (G, b, x, k) be an instance of Collapsed k-Core and let (G', b', x, k) be the instance formed by a graph G' which is obtained from G by adding an universal vertex u, $b' = b + 1$, and x and k from the original instance. We claim that the instances are equivalent.

First if S is a solution for (G, b, x, k), then $S \cup \{u\}$ is a solution for (G', b', x, k), as $G \setminus S = G' \setminus S'$. Second, let S' be a solution for (G', b', x, k). If S' contains u, then $S' \setminus \{u\}$ is a solution for G. Now suppose that S' does not contain u and let v be an arbitrary vertex of S'. We claim that $S'' = (S' \setminus \{v\}) \cup \{u\}$ is also a solution to (G', b', x, k), since $G' \setminus S''$ is isomorphic to a subgraph of $G' \setminus S'$. Indeed, consider the bijection φ, which maps each vertex of $V(G) \setminus S'$ to itself and v to u. To show that it is an isomorphism, it is enough to consider edges incident on v, however, as there is an edge between u and every vertex of $V(G) \setminus S'$, these definitely map to edges. Hence the k-core of $G' \setminus S''$ is at most as large as the k-core of $G' \setminus S'$ and indeed S'' is a solution to (G', b', x, k). Now the equivalence of the instance follows from the case where u is in S'. □

4.1 W[1]-hard Cases

Considering a larger parameter than, e.g., the size of the dominating set, namely the clique cover number[6], we can show W[1]-hardness, even in combination with b. This can be done with a parameterized reduction from MULTICOLORED CLIQUE parameterized by the solution size.

Proposition 3 COLLAPSED k-CORE *is W[1]-hard when parameterized by the combination of b and the clique cover number of the input graph.*

Proof We present a parameterized reduction from the W[1]-hard problem MULTICOLORED CLIQUE parameterized by the solution size [20]. In MULTICOLORED CLIQUE, we are given an integer s and a s-colored graph with color classes $V_1, V_2, \ldots, V_s$, and the task is to find a clique of size s containing one vertex from each color class. Let $(G = (V, E), s, V_1, V_2, \ldots, V_s)$ be an instance of MULTICOLORED CLIQUE. The edge set E can be partitioned into $\binom{s}{2}$ subsets: $E_{i,j} = \{v_z v_y | v_z \in V_i, v_y \in V_j\}, 1 \leq i < j \leq s$. We create an instance $(G' = (V', E'), b, x, k)$ of COLLAPSED k-CORE as follows (see Fig. 8 for an illustration).

- Denote $k = 2\max_{1\leq i<j\leq s} |E_{i,j}|$, $n = |V|$ and set $b = s$, $x = N - N'$ where $N = 2n^4 + k + s + n + k\binom{s}{2}$ is the number of vertices in G' we will construct and $N' = s + k\binom{s}{2}$.
- For every $V_i, i = 1, 2, \ldots, s$, create a clique C_i in G', which contains all vertices in V_i.
- For every $E_{i,j}, 1 \leq i < j \leq s$, create a clique $C_{i,j}$ of size k in G', which contains 2 vertices $v_{z,y}, v'_{z,y}$ for every edge $v_z v_y$ in $E_{i,j}$ and $k - 2|E_{i,j}|$ more dummy vertices.
- For every edge $v_z v_y \in E_{i,j}$, add 4 edges $v_{z,y}v_z$, $v_{z,y}v_y$, $v'_{z,y}v_z$ and $v'_{z,y}v_y$ to G'.
- Create a clique C of size $2n^4 + k + s$. Pick $k + s$ arbitrary vertices in C, and make all these $k + s$ vertices adjacent to all vertices in $\bigcup_{i=1}^{s} C_i$. For every dummy vertex in $\bigcup_{i=1}^{s}\bigcup_{j=i+1}^{s} C_{i,j}$, we add edges between this vertex and two distinct vertices in C. The size of C is large enough such that no pair of edges between $E_{i,j}$ and C share the same endpoint in C.

Notice that the clique cover number of G' is $s + \binom{s}{2} + 1$. We claim that there is a multicolored clique of size s in G if and only if (G', b, x, k) is a yes-instance.

$\Rightarrow$: If there is a multicolored clique S of size s in G, we show in the following that the k-core of $G' - S$ has size at most x. Since $b = s$ and $N' = s + k\binom{s}{2}$, it suffices to show that all edge cliques $C_{i,j}$ collapse. For any clique $C_{i,j}$, every vertex in this clique has degree $k + 1$, since it has $k - 1$ neighbors in $C_{i,j}$ and 2 neighbors in C_i and C_j (or in C for dummy vertices). For any $C_{i,j}$, suppose $v_z, v_y \in S$, where $v_z \in V_i$ and $v_y \in V_j$. Since S is a clique, there is an edge $v_z v_y$ in G, and there are $v_{z,y}$ and

[6]The clique cover number of a graph G is the minimum number of cliques in G such that their union contains all vertices of G.

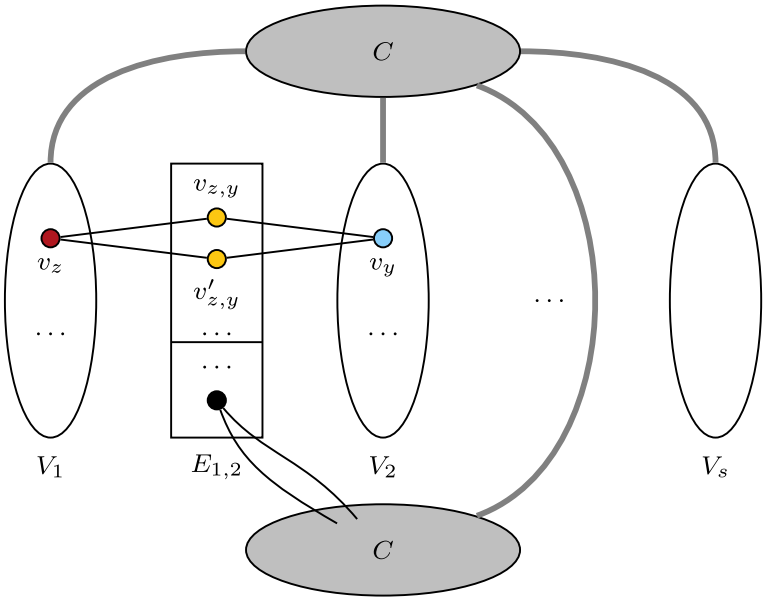

Fig. 8 Illustration of the reduction from MULTICOLORED CLIQUE to COLLAPSED k-CORE with $E_{1,2}$ highlighted. Every gray edge in this figure means that all vertices in one endpoint of this edge are connected to all vertices in the other endpoint. The big clique C is separated into two parts, and the upper part is connected to all V_i with $1 \leq i \leq s$. Two yellow vertices in $E_{1,2}$ represent an edge, and they are connected to the two endpoints of this edge, the red vertex in V_1 and the blue vertex in V_2. Vertices below the line in $E_{1,2}$ are dummy vertices, and each of them is connected to two private vertices in the lower part of C

$v'_{z,y}$, both connected to v_z and v_y in G'. After deleting v_z and v_y, both $v_{z,y}$ and $v'_{z,y}$ collapse, which then causes all remaining vertices in $C_{i,j}$ to collapse. Therefore all edge cliques $C_{i,j}$ will collapse after deleting S.

$\Leftarrow$: Suppose (G', b, x, k) is a yes-instance, we need to show that there is a multicolored clique of size s in G. Let S be the deleted vertex set of size at most b and let S' be the set of all collapsed vertices. Since $N' = s + k\binom{s}{2}$, we have $|S'| \geq k\binom{s}{2}$. Notice that in the subgraph of G' induced by $C \cup \bigcup_{i=1}^{s} C_i$ all vertices in C_i have degree at least $k + s$ and all vertices in C have degree at least $2n^4 + k + s - 1$. Therefore, by Observation 2, these vertices will never collapse and only vertices in $\bigcup_{i=1}^{s} \bigcup_{j=i+1}^{s} C_{i,j}$ can collapse. Since the number of vertices in $\bigcup_{i=1}^{s} \bigcup_{j=i+1}^{s} C_{i,j}$ is exactly $k\binom{s}{2}$, we have that all vertices in $\bigcup_{i=1}^{s} \bigcup_{j=i+1}^{s} C_{i,j}$ will collapse, S only contains vertices from C and $\bigcup_{i=1}^{s} C_i$, and contains exactly s of them.

Suppose S contains t vertices from C and $s - t$ vertices from $\bigcup_{i=1}^{s} C_i$. On one hand, for every clique $C_{i,j}$, the first vertex to collapse must connect to two vertices from S, so overall there must be $2\binom{s}{2}$ edges between all such vertices and S. On the other hand, each vertex in $S \cap C$ can provide at most one such edge and each vertex in S from $\bigcup_{i=1}^{s} C_i$ can provide at most $s - 1$ such edges, so overall the number is strictly less than $2\binom{s}{2}$ if $t > 0$. Since all cliques $C_{i,j}, 1 \leq i < j \leq s$ will collapse, we have $t = 0$ and S only contains vertices from $\bigcup_{i=1}^{s} C_i$.

So the firstly collapsed vertex $v_{z,y}$ in $C_{i,j}$ must connect to two vertices from S, one v_z from C_i and another v_y from C_j. This means S contains exactly one vertex v_i from each C_i and each pair of vertices v_i and v_j connect to at least one common vertex $v_{i,j}$ in $C_{i,j}$, which means v_i and v_j are connected in G. Therefore, S forms a clique of size s in G. □

4.2 Fixed-Parameter Tractable Cases

On the positive side, we sketch a dynamic program on the tree decomposition of the input graph G which implies that COLLAPSED k-CORE is in FPT when parameterized by the treewidth of the input graph.

Proposition 4 COLLAPSED k-CORE *is in FPT when parameterized by the treewidth of the input graph.*

Sketch Let (G, b, x, k) be an instance of COLLAPSED k-CORE. As every graph G is $\mathrm{tw}(G)$-degenerate (see [20, Exercise 7.14]) either $k \leq \mathrm{tw}(G)$ or the k-core of G is (already) empty and we can answer Yes. Hence, for the rest of the proof we assume that $k \leq \mathrm{tw}(G)$. We assume that we are given a nice tree decomposition of G [8, 32] and use dynamic programming on the nice tree decomposition of G. The indices of the table are formed for each bag of the decomposition by the number of vertices of the solution already forgotten, the number of vertices in the core already forgotten, a partition of the bag into three sets B, X, and Q, an (elimination) order for the vertices in Q, and for each vertex in Q the number of its neighbors in X or higher in the order (including those already forgotten). This number is always in $0, \ldots, k-1$, as otherwise it would not be possible to eliminate the vertex.

The set B represents the partial solution (or rather its intersection with the bag), i.e., the vertices to be deleted. The set X represents the vertices which (are free to) remain in the core. The vertices in Q should collapse after removing the vertices of the solution and the collapse of the vertices preceding them in the order.

Morex formally, let V_t be a bag of the decomposition and G_t be the subgraph of the input graph induced by the union of all bags that are descendants of V_t (including V_t itself). Let tuple b', x', B, X, Q, $\preccurlyeq$ and f represent an index of the associated table, that is, $b' \in \{0, \ldots, b\}$, $x' \in \{0, \ldots, x\}$, $B \uplus X \uplus Q$ is a partition of V_t, $\preccurlyeq$ is a total order on Q, and $f : Q \to \{0, \ldots, k-1\}$. Then we set the corresponding table entry to true if and only if there is a partition $B' \uplus X' \uplus Q'$ of $V(G_t)$ and a total order $\trianglelefteq$ on Q' satisfying the following conditions: $B' \cap V_t = B$, $Q' \cap V_t = Q$, $X' \cap V_t = X$, $\preccurlyeq$ is the restriction of $\trianglelefteq$ to Q, $|B' \setminus B| = b'$, $|X' \setminus X| = x'$ and for every $q \in Q'$ we have $k > |\{r \mid \{q, r\} \in E \wedge (r \in X' \vee (r \in Q' \wedge q \trianglelefteq r))\}|$, and, in particular, for $q \in Q$ we have $f(q) = |\{r \mid \{q, r\} \in E \wedge (r \in X' \vee (r \in Q' \wedge q \trianglelefteq r))\}|$.

It is straightforward to compute the content of the table for a particular bag based on the tables for the children bags.

There are $3^{\mathrm{tw}(G)} \cdot \mathrm{tw}(G)^{O(\mathrm{tw}(G))} \cdot k^{\mathrm{tw}(G)} = \mathrm{tw}(G)^{O(\mathrm{tw}(G))}$ possible indices for each bag. Hence the slightly superexponential running time of $\mathrm{tw}(G)^{O(\mathrm{tw}(G))} \cdot n^{O(1)}$ follows. □

Using monadic second order (MSO) logic formulas, it can be shown that for a smaller structural parameter, namely the cliquewidth of the input graph, there are also positive results. Here however, we can only show fixed-parameter tractability for the combination of the cliquewidth of the input graph with k and either b, x, $n - x$, or $n - b$.

Proposition 5 COLLAPSED k-CORE *is in FPT when parameterized by the cliquewidth of the input graph combined with k and either b, x, n − x, or n − b.*

Proof We first develop, for a fixed k a formula

$$\texttt{core}(B, X),$$

which should express that the set X contains the whole vertex set of the k-core of the graph $G - B$. The formula thus says that no graph induced by a set larger than X, but not containing anything from B is a core. In other words, each such graph contains a vertex of degree at most $k - 1$, i.e., not having k distinct neighbors in the set considered. For that purpose we use the following subformula:

$$\begin{aligned}\texttt{smalldeg}_k(v, A) \quad &= \quad \forall x_1 \in V\, \forall x_2 \in V\, \ldots\, \forall x_k \in V \left(\bigwedge_{i=1}^{k} (x_i \in A \wedge \texttt{adj}(v, x_i)) \right) \\ &\Longrightarrow \bigvee_{1 \le i < j \le k} x_i = x_j.\end{aligned}$$

Now the sought formula is

$$\begin{aligned}\texttt{core}(B, X) \quad &= \quad \forall A \subseteq V\, (A \cap B = \emptyset \wedge X \subseteq A \wedge X \neq A) \\ &\Longrightarrow\ \exists v \in V\, (v \in A) \wedge \texttt{smalldeg}_k(v, A).\end{aligned}$$

This formula is of length $O(k^2)$. Combined with some of the following formulae it gives the result for all parameter combinations promised. The following formula bounds a set S passed to be of size at most s:

$$\begin{aligned}\texttt{sizeatmost}_s(S) \quad &= \quad \forall x_1 \in V\, \forall x_2 \in V\, \ldots\, \forall x_s \in V\, \forall x_{s+1} \in V\, (\bigwedge_{i=1}^{s+1} x_i \in S) \\ &\Longrightarrow \bigvee_{1 \le i < j \le s+1} x_i = x_j.\end{aligned}$$

The next formula bounds the size to at most $n - s$:

$$\begin{aligned}&\texttt{sizeatmost}_{n-s}(S) = \\ &\exists x_1 \in V\, \exists x_2 \in V\, \ldots\, \exists x_s \in V \left(\bigwedge_{i=1}^{s} x_i \notin S \right) \wedge \left(\bigwedge_{1 \le i < j \le s} x_i \neq x_j \right).\end{aligned}$$

Both these formulae have length $O(s^2)$.

Now the result follows from the theorem of Courcelle et al. [18] as it allows for optimization over the size of one free set variable. Hence, the result for parameterization by the cliquewidth, k, and b is obtained by minimizing the size of X

satisfying $\exists B \subseteq V\ \texttt{core}(B, X) \wedge \texttt{sizeatmost}_b(B)$ in the input graph, where the formula is of size $O(k^2 + b^2)$. The result for parameterization by the cliquewidth, k, and x is obtained by minimizing the size of B satisfying $\exists X \subseteq V\ \texttt{core}(B, X) \wedge \texttt{sizeatmost}_x(X)$, where the formula is of size $O(k^2 + x^2)$. The result for parameterization by the cliquewidth, k, and $n - x$ is obtained by minimizing the size of B satisfying $\exists X \subseteq V\ \texttt{core}(B, X) \wedge \texttt{sizeatmost}_{n-(n-x)}(X)$, where the formula is of size $O(k^2 + (n - x)^2)$. Finally, the result for parameterization by the cliquewidth, k, and $n - b$ is obtained by minimizing the size of X satisfying $\exists B \subseteq V\ \texttt{core}(B, X) \wedge \texttt{sizeatmost}_{n-(n-b)}(B)$ in the input graph, where the formula is of size $O(k^2 + (n - b)^2)$. □

4.3 Kernelization Lower Bounds

In the final subsection of this section, we show that Collapsed k-Core does not admit a polynomial kernel when parameterized by rather large parameter combinations.

We employ the OR-cross-composition framework by [7] to refute the existence of a polynomial kernel for a parameterized problem under the assumption that $\mathsf{NP} \not\subseteq \mathsf{coNP}/poly$, the negation of which would cause a collapse of the polynomial-time hierarchy to the third level. Informally, in an OR-cross-composition, we have to *compose* many problem instances of an NP-hard problem into one big instance of the problem we want to investigate. This composition should then have the property that the big instance is a YES-instance if and only if at least one of the input instances is a YES-instance. This then refutes polynomial kernels for the problem under investigation when parameterized by any parameter that only depend on the maximum size of the input instances and only logarithmically on the number of input instances. In order to formally describe the framework, we need some definitions first.

An equivalence relation $\mathcal{R}$ on the instances of some problem L is a *polynomial equivalence relation* if

1. one can decide for each two instances in time polynomial in their sizes whether they belong to the same equivalence class, and
2. for each finite set S of instances, $\mathcal{R}$ partitions the set into at most $(\max_{x \in S} |x|)^{O(1)}$ equivalence classes.

Using this, we can now define OR-cross-compositions.

An *OR-cross-composition* of a problem $L \subseteq \Sigma^*$ into a parameterized problem P (with respect to a polynomial equivalence relation $\mathcal{R}$ on the instances of L) is an algorithm that takes T $\mathcal{R}$-equivalent instances $x_1, \ldots, x_T$ of L and constructs in time polynomial in $\sum_{i=1}^{T} |x_i|$ an instance (x, k) of P such that

1. k is polynomially upper-bounded in $\max_{1 \le i \le T} |x_i| + \log(T)$ and
2. (x, k) is a YES-instance of P if and only if there is an $i \in [T]$ such that x_i is a YES-instance of L.

If an NP-hard problem L OR-cross-composes into a parameterized problem P, then P does not admit a polynomial kernel, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ [7].

We first show an OR-cross composition from Cubic Vertex Cover.

Theorem 4 *For all $k \geq 2$* COLLAPSED k-CORE *does not admit a polynomial kernel when parameterized by the combination of b and the vertex cover number of the input graph unless NP $\subseteq$ coNP/poly.*

Proof We apply an OR-cross composition from the NP-hard problem CUBIC VERTEX COVER [26]. In CUBIC VERTEX COVER, we are given a 3-regular graph G and an integer s, and the task is to find a vertex subset of size at most s which contains at least one endpoint of each edge of G.

We say an instance of CUBIC VERTEX COVER is *malformed* if the string does not represent a pair (G, s), where G is a 3-regular graph and s is a positive integer. It is *trivial*, if $s \geq |V(G)|$. We define the equivalence relation $\mathcal{R}$ as follows: all malformed instances are equivalent, all trivial instances are equivalent and two well-formed non-trivial instances (G, s) and (G', s') are $\mathcal{R}$-equivalent if $|V(G)| = |V(G')|$ and $s = s'$. Observe that $\mathcal{R}$ is a polynomial equivalence relation.

Let the input consist of T $\mathcal{R}$-equivalent instances of CUBIC VERTEX COVER. If the instances are malformed or trivial, we return a constant size no- or yes-instance of COLLAPSED k-CORE, respectively. Let $(G_i, s)_{0 \leq i \leq T-1}$ be well-formed non-trivial $\mathcal{R}$-equivalent instances of CUBIC VERTEX COVER. Since all instances have the same size of the vertex set, we can assume they share the same vertex set $V = \{v_1, v_2, \ldots, v_n\}$. We assume T to be a power of 2, as otherwise we can duplicate some instances. Now we create an instance (G, b, x, k) of COLLAPSED k-CORE for some arbitrary but fixed $k \geq 2$ as follows.

- Set $b = s + 2ks \log T$ and $x = N - N'$, where $N = n + \frac{3}{2}nT + 4ks \log T + k + s + \frac{3}{2}n(k-2)$ is the number of all vertices in graph G we will construct and $N' = s + 2ks \log T + \frac{3}{2}n$. (Note that $\frac{3}{2}n$ is integer since each G_i is a 3-regular graph and this is the number of its edges.)
- First for every vertex v_p in V, create a vertex v_p in G.
- For all $0 \leq i \leq T-1$ and for every edge set $E(G_i)$, create a vertex set V_i^E in G, in which a vertex $v_{p,q}$ represents an edge $v_p v_q$ in $E(G_i)$. Then we have T of these vertex sets and each set has $\frac{3}{2}n$ vertices.
- For every edge $v_p v_q$ in $E(G_i)$, add 2 edges $v_{p,q} v_p$ and $v_{p,q} v_q$ in G.
- Now create the selection gadget in G. It contains $\log T$ pairs of cliques C_j^d $(0 \leq j \leq \log T - 1, d \in \{0, 1\})$, and all of them have the same size of $2ks$. For every vertex set V_i^E, let $i = (d_{\log T-1} d_{\log T-2} \ldots d_0)_2$ be the binary representation of the index i, where $d_j \in \{0, 1\}$ for $0 \leq j \leq \log T - 1$ and we add leading zeros so that the length of the representation is exactly $\log T$. We add edges between all vertices in V_i^E and all vertices in $\bigcup_{j=0}^{\log T-1} C_j^{d_j}$.
- Finally we create a clique C with $|C| = k + b + \frac{3}{2}n(k-2)$, which contains two parts of vertices. The first part contains $k + b$ vertices and each of them connects to all vertices in $V \cup \bigcup_{j=0}^{\log T-1} \bigcup_{d=0}^{1} C_j^d$. The second part of $\frac{3}{2}n(k-2)$ vertices is connected to vertices in V_i^E for all $0 \leq i \leq T-1$ in the following way. For every vertex $v_{p,q}$ in V_i^E, add edges between $v_{p,q}$ and $k-2$ vertices in C. We make sure that all vertices in the same V_i^E connect to different vertices in C. In

other words, every vertex in the second part of C connects to exactly one vertex in every V_i^E.

Notice that the vertex cover number of G is at most $n+4ks\log T+k+b+\frac{3}{2}n(k-2)$ as $V \cup \bigcup_{j=0}^{\log T-1} \bigcup_{d=0}^{1} C_j^d \cup C$ forms a vertex cover of the graph. The construction is illustrated in Fig. 9. We now show that at least one instance (G_i, s) is a yes-instance if and only if the instance (G, b, x, k) of Collapsed k-Core constructed above is a yes-instance.

$\Rightarrow$: If (G_i, s) is a yes-instance, which means that there is a vertex subset V^* of size s that covers all edges in G_i, then we delete the corresponding s vertices in G and all vertices in $\bigcup_{j=0}^{\log T-1} C_j^{d_j}$, where $i = (d_{\log T-1} \dots d_0)_2$ is the binary representation of i. So far, we deleted $s + 2ks\log T$ vertices, and all vertices in V_i^E will collapse, since they just have at most $k-1$ edges remaining, $k-2$ of which connect to vertices in C and at most one to vertices in V. Therefore, the number of remaining vertices is x and instance (G, b, x, k) is a yes-instance.

$\Leftarrow$: If (G, b, x, k) is a yes-instance, we need to show that there is at least one instance which has a vertex cover of size at most s. Let S be the set of deleted vertices of size at most b and let S' be the set of all collapsed vertices. Since $N' = s + 2ks\log T + \frac{3}{2}n$, we have $|S'| \geq \frac{3}{2}n$. In the subgraph of G induced by V, C and

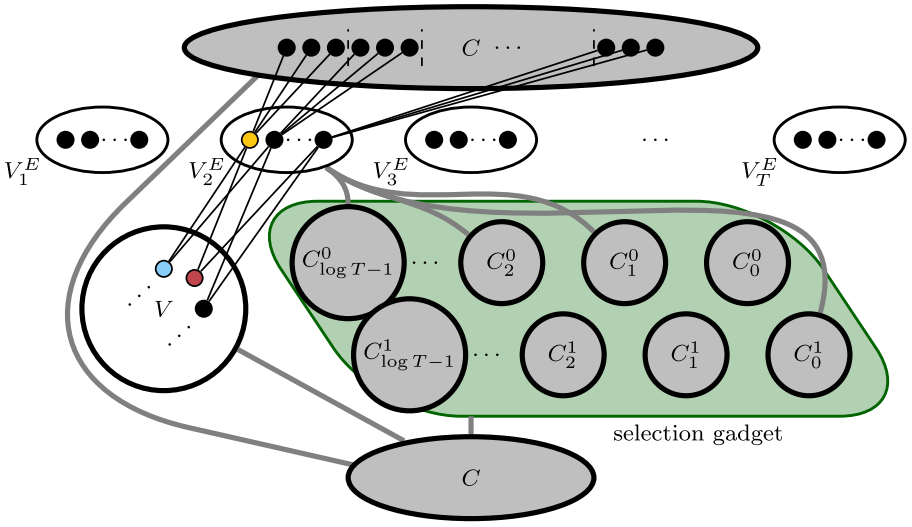

Fig. 9 Illustration of the OR-cross composition from Cubic Vertex Cover to Collapsed k-Core with $k = 5$. The selection gadget consists of all the circles contained in the green box. Every gray edge in this figure means that all vertices in one endpoint of this edge are connected to all vertices in the other endpoint. Every vertex in V_i^E connects to two endpoints of its corresponding edge in G_i. For example, the yellow vertex in V_2^E is connected to the blue and the red vertex in V, which represents the two endpoints of the corresponding edge in G_2. The big clique C is separated into two parts. Every vertex in V_i^E is connected to $k-2$ vertices in the upper part of C. Since $k = 5$, the yellow vertex in V_2^E is connected to three vertices in C. Vertices contained in thick outlined vertex sets form a vertex cover. To keep the picture simple, edges that contain vertices from V_i^E with $i \neq 2$ are not depicted

$\bigcup_{j=0}^{\log T-1}\bigcup_{d=0}^{1} C_j^d$ all vertices in $V \cup \bigcup_{j=0}^{\log T-1}\bigcup_{d=0}^{1} C_j^d \cup C$ have degree larger than $k+b$. Hence, by Observation 2 they will not collapse and all collapsed vertices come from $\bigcup_{i=0}^{T-1} V_i^E$.

We show that all collapsed vertices can only come from one single V_i^E for some i. Suppose two vertices v and v' from different sets of V_i^E $(0 \le i \le T-1)$ collapse after deleting S, then there is at least one pair of cliques $C_{j_0}^0$ and $C_{j_0}^1$ such that v is connected to all vertices in $C_{j_0}^{d_0}$ for some $d_0 \in \{0,1\}$ and v' is connected to all vertices in $C_{j_0}^{1-d_0}$. To make v collapse, at least $2ks\log T-(k-1)$ vertices from the corresponding cliques in the selection gadget need to be deleted. Then to make v' collapse, at least $2ks-(k-1)$ vertices from $C_{j_0}^{1-d_0}$ need to be deleted. Therefore, at least $2ks\log T+2ks-2(k-1)$ vertices need to be deleted, which is strictly more than b. This means that the collapsed vertices come from one single V_i^E. Since $|S'| \ge \frac{3}{2}n$ and $|V_i^E| = \frac{3}{2}n$, we have $S' = V_i^E$ and $S \cap V_i^E = \emptyset$ for some i.

We consider the vertex set S. We know that after deleting S, all vertices in V_i^E collapse. Denote V_I the vertex set of all vertices in $\bigcup_{j=0}^{\log T-1} C_j^{d_j}$, where $i = (d_{\log T-1}\ldots d_0)_2$ is the binary representation of i. Since every vertex in V_I is connected to all vertices in V_i^E, to make V_i^E collapse, it is always better to choose vertices from V_I than any other vertex. If S does not contain all vertices from V_I, we can update S by replacing any $|V_I \setminus S|$ vertices in S with vertices in $V_I \setminus S$. Then $V_I \subseteq S$.

Suppose there is a vertex $v_{p,q}$ in V_i^E such that both v_p and v_q are not in $S \cap V$, then S contains at least one vertex v_c in C connected to $v_{p,q}$, as otherwise $v_{p,q}$ has degree at least k and will not collapse. We update S by replacing v_c with v_p. This will not influence the size of S and more importantly, this will not influence the collapsed set $S' = V_i^E$, since v_c in C is connected to only one vertex $v_{p,q}$ in V_i^E, and $v_{p,q}$ will still collapse under the new S. By updating S in the same way for other vertices in V_i^E not covered by vertices in $S \cap V$, we get a vertex set $S \cap V$ which covers all vertices in V_i^E at least once. And $|S \cap V| \le s$, since $V_I \subseteq S$ and $|V_I| = 2ks\log T$. This corresponds to a vertex cover of size s in G_i. □

Lastly, we show that there is a simple OR-cross composition [7, 20] from Collapsed k-Core onto itself. Note that the parameter combination of the following result is incomparable to that of Theorem 4.

Proposition 6 *For all $k \ge 1$ Collapsed k-Core does not admit a polynomial kernel when parameterized by the combination of b, $n-x$, and the bandwidth of the input graph unless NP $\subseteq$ coNP/poly.*

Proof We apply an OR-cross composition from Collapsed k-Core to Collapsed k-Core.

We say an instance of Collapsed k-Core is *malformed* if the string does not represent a quadruple (G, b, x, k), where G is a graph and b, x are non-negative integers. It is *trivial*, if $b \ge |V(G)|$, $x \ge |V(G)|$, or $k \ge |V(G)|$. We define the equivalence relation $\mathcal{R}$ as follows: all malformed instances are equivalent, all trivial instances are equivalent, and two well-formed non-trivial instances (G_1, b_1, x_1, k)

and (G_2, b_2, x_2, k) are $\mathcal{R}$-equivalent if $|V(G_1)| = |V(G_2)|$, $b_1 = b_2$ and $x_1 = x_2$. Observe that $\mathcal{R}$ is a polynomial equivalence relation.

Let the input consist of T $\mathcal{R}$-equivalent instances of Collapsed k-Core. If the instances are malformed or trivial, we return a constant size no- or yes- instance of Collapsed k-Core, respectively. Let $(G_i, b_i, x_i, k)_{1 \leq i \leq T}$ be well-formed non-trivial $\mathcal{R}$-equivalent instances of Collapsed k-Core. Since all instances have the same b_i, x_i and all G_i with $1 \leq i \leq T$ have the same size of vertex set, we denote $b = b_i, x = x_i$ and $n = |V(G_i)|$. If $n \leq 3$, then we solve all the instances in $O(T)$ time and output a constant-size instance with the appropriate answer.

Now we create an instance (G, b', x', k) of Collapsed k-Core. We start by making G a disjoint union of all G_i. For each $i \in \{1, \ldots, T\}$, we add to G two cliques C_i and C_i', each of size n^2, and add all edges between G_i and C_i and between C_i and C_i'. Note that G contains $T(n + 2n^2)$ vertices in total. Set $b' = b + n^2$ and $x' = (n + 2n^2)(T - 1) + n^2 + x$.

Notice that the bandwidth [13] of G is upper bounded by $2n^2$ and $|V(G)| - x' = T(n + 2n^2) - (T - 1)(n + 2n^2) - n^2 - x \leq n^2 + n$. We now show that at least one instance (G_i, b, x, k) is a yes-instance if and only if the instance (G, b', x', k) of Collapsed k-Core constructed above is a yes-instance.

$\Rightarrow$: If (G_i, b, x, k) is a yes-instance, then there is a subset $S \subseteq V(G_i)$ such that the k-core of $G_i - S$ has size at most x. If we delete all vertices from S and the whole clique C_i in G, then at most x vertices remain in the k-core of $G_i - S$. Therefore the k-core of $G - S - V(C_i)$ has size at most x'.

$\Leftarrow$: If (G, b', x', k) is a yes-instance, then let S be the set of deleted vertices of size at most b' and let S' be the set of all collapsed vertices. Since the degree of vertices in C_i and C_i' is at least $2n^2 - 1$ which is more than $n^2 + 2n \geq b' + k$ for $n \geq 3$, these vertices will never collapse by Observation 2. So S' only contains vertices from $\bigcup_{i=1}^{T} V(G_i)$. Furthermore, it is impossible for two vertices v_i and v_j from different sets $V(G_i)$ and $V(G_j)$ to collapse after deleting S. Indeed, suppose they do collapse, then $|S \cap C_i| \geq n^2 - k$ and $|S \cap C_j| \geq n^2 - k$, which means $|S| \geq 2n^2 - 2k \geq 2n^2 - 2n \geq n^2 + n > n^2 + b = b'$, where the middle inequality follows from $n \geq 3$. Therefore S' only contains vertices from a single graph, say G_i.

Since G contains $T(n+2n^2)$ vertices in total, $x' = (n+2n^2)(T-1)+n^2+x$, and $b' = b + n^2$, we have $|S'| \geq n - b - x$. To make vertices in G_i collapse, it is always better to choose vertices from C_i into S, as vertices from C_i connect to all vertices in G_i. Thus we can assume $C_i \subseteq S$. Then $|S \cap V(G_i)| \leq b$. If $|V(G_i) \setminus (S \cup S')| > x$, which can only happen if $|S \cap V(G_i)| < b$, then we can remove vertices from $S \setminus (C_i \cup V(G_i))$ and add vertices from $G_i \setminus (S \cup S')$ to S until $|S \cap V(G_i)| = b$. This will not influence the collapsed vertices in S'. Then we get a vertex set $S_i = S \cap V(G_i)$, and the k-core of $G_i - S_i$ has size at most x. □

5 Conclusion

Our results highlight a dichotomy in the computational complexity of Collapsed k-Core for $k \leq 2$ and $k \geq 3$. Along the way, we correct some inaccuracies in the literature concerning the parameterized complexity of Collapsed k-Core with $k = 3$

and $x = 0$ and give a simple single exponential linear time parameterized algorithm for COLLAPSED k-CORE with $k = 2$, which almost matches the simplest known, independently found, single exponential linear time algorithm for FEEDBACK VERTEX SET. We leave as an open question whether COLLAPSED k-CORE with $k = 2$ on graphs of maximum degree 3 is solvable in polynomial time if there are no forbidden vertices. We further investigate the parameterized complexity with respect to several structural parameters of the input graph. As a highlight we show that COLLAPSED k-CORE does not admit polynomial kernels for rather large parameter combinations. We leave the complexity of COLLAPSED k-CORE when parameterized solely by the cliquewidth of the input graph open.

Acknowledgements This research was initiated at the annual research retreat of the Algorithmics and Computational Complexity (AKT) group of TU Berlin, held in Darlingerode, Germany, from March 19 till March 23, 2018. The authors would like to thank Anne-Sophie Himmel for initial discussions leading to the results in this paper and to the anonymous referees of IPEC and this journal for detailed comments that helped to significantly improve the presentation of the paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Abrahamson, K.A., Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness IV: on completeness for w[P] and PSPACE analogues. Annals Pure Appl. Logic **73**(3), 235–276 (1995)
2. Batagelj, V., Zaveršnik, M.: Fast algorithms for determining (generalized) core groups in social networks. Adv. Data Anal. Classif. **5**(2), 129–145 (2011)
3. Bazgan, C., Chopin, M.: The complexity of finding harmless individuals in social networks. Discret. Optim. **14**, 170–182 (2014)
4. Bazgan, C., Chopin, M., Nichterlein, A., Sikora, F.: Parameterized inapproximability of target set selection and generalizations. Computability **3**(2), 135–145 (2014)
5. Ben-Zwi, O., Hermelin, D., Lokshtanov, D., Newman, I.: Treewidth governs the complexity of target set selection. Discret. Optim. **8**(1), 87–96 (2011)
6. Bhawalkar, K., Kleinberg, J., Lewi, K., Roughgarden, T., Sharma, A.: Preventing unraveling in social networks: the anchored k-core problem. SIAM J. Discret. Math. **29**(3), 1452–1475 (2015)
7. Bodlaender, H.L., Jansen, B.M., Kratsch, S.: Kernelization lower bounds by cross-composition. SIAM J. Discret. Math. **28**(1), 277–305 (2014)
8. Bodlaender, H.L., Drange, P.G., Dregi, M.S., Fomin, F.V., Lokshtanov, D., Pilipczuk, M.: A $c^k n$ 5-approximation algorithm for treewidth. SIAM J. Comput. **45**(2), 317–378 (2016)
9. Cao, Y.: A naive algorithm for feedback vertex set. In: Proceedings of the 1st symposium on simplicity in algorithms, (SOSA '18), Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, OASICS, vol. 61, pp. 1:1–1:9 (2018)
10. Cao, Y., Chen, J., Liu, Y.: On feedback vertex set: New measure and new structures. Algorithmica **73**(1), 63–86 (2015)

11. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. Theor. Comput. Sci. **411**(40-42), 3736–3756 (2010)
12. Chen, N.: On the approximability of influence in social networks. SIAM J. Discret. Math. **23**(3), 1400–1415 (2009)
13. Chinn, P.Z., Chvátalová, J., Dewdney, A.K., Gibbs, N.E.: The bandwidth problem for graphs and matrices—a survey. J. Graph Theor. **6**(3), 223–254 (1982)
14. Chitnis, R., Talmon, N.: Can we create large k-cores by adding few edges? In: Computer Science - Theory and Applications - Proceedings of the 13th International Computer Science Symposium in Russia (CSR 2018), Springer, Lecture Notes in Computer Science, vol. 10846, pp. 78–89 (2018)
15. Chitnis, R., Fomin, F.V., Golovach, P.A.: Parameterized complexity of the anchored k-core problem for directed graphs. Inf. Comput. **247**, 11–22 (2016)
16. Chitnis, R.H., Fomin, F.V., Golovach, P.A.: Preventing unraveling in social networks gets harder. In: Proceedings of the 27th AAAI conference on artificial intelligence (AAAI '13), pp. 1085–1091. AAAI Press (2013)
17. Chopin, M., Nichterlein, A., Niedermeier, R., Weller, M.: Constant thresholds can make target set selection tractable. Theor. Comput. Syst. **55**(1), 61–83 (2014)
18. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. Theor. Comput. Syst. **33**(2), 125–150 (2000)
19. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: Proceedings of the IEEE 52nd annual symposium on foundations of computer science (FOCS '11), pp. 150–159. IEEE Computer Society (2011)
20. Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer, New York (2015)
21. Diestel, R.: Graph Theory, 5th edn, Graduate Texts in Mathematics, vol. 173. Springer, New York (2016)
22. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Springer, New York (2013)
23. Dvořák, P., Knop, D., Toufar, T.: Target set selection in dense graph classes. In: Proceedings of the 29th international symposium on algorithms and computation (ISAAC '18), Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, LIPIcs, vol. 123, pp. 18:1–18:13 (2018)
24. Fomin, F.V., Kratsch, S., Pilipczuk, M., Pilipczuk, M., Villanger, Y.: Tight bounds for parameterized complexity of cluster editing with a small number of clusters. J. Comput. Syst. Sci. **80**(7), 1430–1447 (2014)
25. Garcia, D., Mavrodiev, P., Schweitzer, F.: Social resilience in online communities: The autopsy of friendster. In: Proceedings of the 1st ACM conference on online social networks (COSN '13), pp. 39–50. ACM (2013)
26. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman (1979)
27. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. J. Comput. Syst. Sci. **72**(8), 1386–1396 (2006)
28. Hartmann, T.A.: Target set selection parameterized by clique-width and maximum threshold. In: Proceedings of the 44th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2018), Springer, Lecture Notes in Computer Science, vol. 10706, pp. 137–149 (2018)
29. Iwata, Y.: Linear-time kernelization for feedback vertex set. In: 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Leibniz International Proceedings in Informatics (LIPIcs), vol. 80, pp. 68:1–68:14 (2017)
30. Iwata, Y., Kobayashi, Y.: Improved analysis of highest-degree branching for feedback vertex set. In: Proceedings of the 14th International Symposium on Parameterized and Exact Computation (IPEC '19), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Leibniz International Proceedings in Informatics (LIPIcs), vol. 148, pp. 22:1–22:11 (2019)
31. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. Theor. Comput. **11**, 105–147 (2015)
32. Kloks, T.: Treewidth, Computations and Approximations, Lecture Notes in Computer Science, vol. 842. Springer, New York (1994)

33. Kociumaka, T., Pilipczuk, M.: Faster deterministic feedback vertex set. Inf. Process. Lett. **114**(10), 556–560 (2014)
34. Lokshtanov, D., Marx, D., Saurabh, S.: Lower bounds based on the exponential time hypothesis. Bull. EATCS 3(105) (2013)
35. Lokshtanov, D., Ramanujan, M., Saurabh, S.: Linear time parameterized algorithms for subset feedback vertex set. ACM Trans. Algo. (TALG) **14**(1), 7 (2018)
36. Luo, J., Molter, H., Suchý, O.: A parameterized complexity view on collapsing k-Cores. In: 13th International Symposium on Parameterized and Exact Computation (IPEC 2018), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Leibniz International Proceedings in Informatics (LIPIcs), vol. 115, pp. 7:1–7:14 (2019)
37. Malliaros, F.D., Vazirgiannis, M.: To stay or not to stay: modeling engagement dynamics in social graphs. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13), pp. 469–478. ACM (2013)
38. Mathieson, L.: The parameterized complexity of editing graphs for bounded degeneracy. Theor. Comput. Sci. **411**(34-36), 3181–3187 (2010)
39. Mathieson, L., Szeider, S.: The parameterized complexity of regular subgraph problems and generalizations. In: Proceedings of the 14th Computing: the Australasian theory symposium (CATS '08), pp 79–86 (2008)
40. Nichterlein, A., Niedermeier, R., Uhlmann, J., Weller, M.: On tractable cases of target set selection. Soc. Netw. Anal. Min. **3**(2), 233–256 (2013)
41. Peleg, D.: Immunity against Local Influence. In: Language, Culture, Computation. Computing - Theory and Technology - Essays Dedicated to Yaacov Choueka on the Occasion of His 75Th Birthday, Part I, Springer, Lecture Notes in Computer Science, vol. 8001, pp. 168–179 (2014)
42. Seidman, S.B.: Network structure and minimum degree. Soc. Networks **5**(3), 269–287 (1983)
43. Ueno, S., Kajitani, Y., Gotoh, S.: On the nonseparating independent set problem and feedback set problem for graphs with no vertex degree exceeding three. Discret. Math. **72**(1-3), 355–360 (1988)
44. Wang, X., Donaldson, R., Nell, C., Gorniak, P., Ester, M., Bu, J.: Recommending groups to users using user-group engagement and time-dependent matrix factorization. In: Proceedins of the 30th AAAI conference on artificial intelligence (AAAI '16), pp. 1331–1337. AAAI Press (2016)
45. Wu, S., Das Sarma, A., Fabrikant, A., Lattanzi, S., Tomkins, A.: Arrival and departure dynamics in social networks. In: Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM '13), pp. 233–242. ACM (2013)
46. Zhang, F., Zhang, Y., Qin, L., Zhang, W., Lin, X.: Finding critical users for social network engagement: The collapsed k-core problem. In: Proceedins of the 31st AAAI conference on artificial intelligence (AAAI '17), pp. 245–251. AAAI Press (2017)

Affiliations

Junjie Luo[1,2,3] · Hendrik Molter[1] · Ondřej Suchý[4]

Hendrik Molter
h.molter@tu-berlin.de

Ondřej Suchý
ondrej.suchy@fit.cvut.cz

1 Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany

2 Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

3 School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, China

4 Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic