Covering Small Independent Sets and Separators with Applications to Parameterized Algorithms*

DANIEL LOKSHTANOV, University of Bergen, Norway
FAHAD PANOLAN, University of Bergen, Norway
SAKET SAURABH, Institute of Mathematical Sciences, India
ROOHANI SHARMA, Institute of Mathematical Sciences, India
MEIRAV ZEHAVI, Ben-Gurion University, Israel

We present two new combinatorial tools for the design of parameterized algorithms. The first is a simple linear time randomized algorithm that given as input a d-degenerate graph G and an integer k, outputs an independent set Y, such that for every independent set X in G of size at most k, the probability that X is a subset of Y is at least $\left(\binom{(d+1)k}{k} \cdot k(d+1)\right)^{-1}$. The second is a new (deterministic) polynomial time graph sparsification procedure that given a graph G, a set $T = \{\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_\ell, t_\ell\}\}$ of terminal pairs and an integer k, returns an induced subgraph $G^\star$ of G that maintains *all* the inclusion minimal multicuts of G of size at most k, and does not contain any $(k+2)$-vertex connected set of size $2^{\mathcal{O}(k)}$. In particular, $G^\star$ excludes a clique of size $2^{\mathcal{O}(k)}$ as a topological minor. Put together, our new tools yield new randomized fixed parameter tractable (FPT) algorithms for Stable s-t Separator, Stable Odd Cycle Transversal and Stable Multicut on general graphs, and for Stable Directed Feedback Vertex Set on d-degenerate graphs, resolving two problems left open by Marx et al. [ACM Transactions on Algorithms, 2013]. All of our algorithms can be derandomized at the cost of a small overhead in the running time.

CCS Concepts: • **Theory of computation** → **Fixed parameter tractability**;

Additional Key Words and Phrases: Independece covering family, Stable Multicut, Stable s-t separator, Stable OCT, Parameterized algorithms

ACM Reference Format:
Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. 2018. Covering Small Independent Sets and Separators with Applications to Parameterized Algorithms. *ACM Comput. Entertain.* 9, 4, Article 39 (March 2018), 37 pages. https://doi.org/0000001.0000001

*Supported by Pareto-Optimal Parameterized Algorithms, ERC Starting Grant 715744, Parameterized Approximation, ERC Starting Grant 306992, and Rigorous Theory of Preprocessing, ERC Advanced Investigator Grant 267959.

Authors' addresses: Daniel Lokshtanov, University of Bergen, Bergen, Norway, daniello@ii.uib.no; Fahad Panolan, University of Bergen, Bergen, Norway, fahad.panolan@ii.uib.no; Saket Saurabh, Institute of Mathematical Sciences, Chennai, Tamil Nadu, India, saket@imsc.res.in; Roohani Sharma, Institute of Mathematical Sciences, Chennai, Tamil Nadu, India, roohani@imsc.res.in; Meirav Zehavi, Ben-Gurion University, Beersheva, Israel, meiravze@bgu.ac.il.

1544-3574/2018/3-ART39 $15.00
https://doi.org/0000001.0000001

1 INTRODUCTION

We present two new combinatorial tools for designing parameterized algorithms. The first is a simple linear time randomized algorithm that given as input a d-degenerate graph G and an integer k, outputs an independent set Y, such that for every independent set X in G of size at most k, the probability that X is a subset of Y is at least $\left(\binom{k(d+1)}{k} \cdot k(d+1)\right)^{-1}$. Here, an *independent set* in a graph G is a vertex set X such that no two vertices in X are connected by an edge, and the *degeneracy* of an n-vertex graph G is the minimum integer d such that there exists an ordering $\sigma : V(G) \to \{1, \ldots, n\}$ such that every vertex v has at most d neighbors u with $\sigma(u) > \sigma(v)$. Such an ordering σ is called a d-*degeneracy sequence* of G. We say that a graph is d-degenerate, if G has a d-degeneracy sequence. More concretely, we prove the following result.

Lemma 1.1. *There exists a linear time randomized algorithm that given as input a d-degenerate graph G and an integer k, outputs an independent set Y, such that for every independent set X in G of size at most k the probability that X is a subset of Y is at least $\left(\binom{k(d+1)}{k} \cdot k(d+1)\right)^{-1}$.*

Proof. Given G, k and a d-degeneracy sequence σ of G the algorithm sets $p = \frac{1}{d+1}$ and colors the vertices of G black or white independently with the following probability : a vertex gets color black with probability p and white with probability $1 - p$. The algorithm then constructs the set Y which contains every vertex v that is colored black and all the neighbors u of v with $\sigma(u) > \sigma(v)$ are colored white. We first show that Y is an independent set. Suppose not. Let $u, v \in Y$, such that $\sigma(u) < \sigma(v)$ and $uv \in E(G)$. Since $u \in Y$, by the construction of Y, v has to be colored white. This contradicts that $v \in Y$ because every vertex in Y is colored black.

We now give a lower bound on the probability with which a given independent set X of size at most k is contained in Y. Define Z to be the set of vertices u such that u has a neighbor $x \in X$ with $\sigma(x) < \sigma(u)$. Since every $x \in X$ has at most d neighbors u with $\sigma(x) < \sigma(u)$, it follows that $|Z| \leqslant kd$. Observe that $X \subseteq Y$ precisely when all the vertices in X are colored black and all the vertices in Z are colored white. This happens with probability

$$p^{|X|}(1-p)^{|Z|} \geqslant \left(\frac{k}{k(d+1)}\right)^{k} \cdot \left(\frac{kd}{k(d+1)}\right)^{kd} \geqslant \left[\binom{(d+1)k}{k} \cdot k(d+1)\right]^{-1}.$$

Here, the last inequality follows from the fact that binomial distributions are centered around their expectation. This concludes the proof. □

Lemma 1.1 allows us to reduce many problems with an independence constraint to the same problem without the independence requirement. For an example, consider the following four well-studied problems:

- Minimum s-t Separator: Here, the input is a graph G, an integer k and two vertices s and t, and the task is to find a set S of at most k vertices such that s and t are in distinct connected components of $G - S$. This is a classic problem solvable in polynomial time [Ford and Fulkerson 1956; Stoer and Wagner 1997].
- Odd Cycle Transversal: Here, the input is a graph G and an integer k, and the task is to find a set S of at most k vertices such that $G - S$ is bipartite. This problem is NP-complete [Choi et al. 1989] and has numerous fixed-parameter tractable (FPT) algorithms [Lokshtanov et al. 2014; Reed et al. 2004]. For all our purposes,

the $\mathcal{O}(4^k \cdot k^{\mathcal{O}(1)} \cdot (n+m))$ time algorithms of Iwata et al. [Iwata et al. 2014] and Ramanujan and Saurabh [Ramanujan and Saurabh 2014] are the most relevant.

- MULTICUT: Here, the input is a graph G, a set $T = \{\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_\ell, t_\ell\}\}$ of terminal pairs and an integer k, and the task is to find a set S on at most k vertices such that for every $i \leqslant \ell$, s_i and t_i are in distinct connected components of $G - S$. Such a set S is called a *multicut* of T in G. This problem is NP-complete even for 3 terminal pairs, that is, when $l = 3$ [Dahlhaus et al. 1994], but it is FPT [Bousquet et al. 2011; Marx and Razgon 2014] parameterized by k, admitting an algorithm [Lokshtanov et al. 2016a] with running time $2^{\mathcal{O}(k^3)} \cdot mn \log n$.
- DIRECTED FEEDBACK VERTEX SET: Here, the input is a directed graph D and an integer k, and the task is to find a set S on at most k vertices such that $D - S$ is acyclic. This problem is also NP-complete [Karp 1972] and FPT [Chen et al. 2008] parameterized by k, admitting an algorithm [Lokshtanov et al. 2016a] with running time $\mathcal{O}(k! \cdot 4^k \cdot k^5 \cdot (n+m))$.

In the "stable" versions of all of the above-mentioned problems, the solution set S is required to be an independent set[2]. Fernau [Demaine et al. 2007] posed as an open problem whether STABLE ODD CYCLE TRANSVERSAL is FPT. This problem was resolved by Marx et al. [Marx et al. 2013], who gave FPT algorithms for STABLE s-t SEPARATOR running in time $2^{2^{k^{O(1)}}} \cdot (n+m)$ and STABLE ODD CYCLE TRANSVERSAL running in time $2^{2^{k^{O(1)}}} \cdot (n+m) + \mathcal{O}(3^k \cdot nm)$. Here, the $\mathcal{O}(3^k \cdot nm)$ term in the running time comes from a direct invocation of the algorithm of Reed et al. [Reed et al. 2004] for ODD CYCLE TRANSVERSAL. Furthermore, Marx et al. [Marx et al. 2013] gave an algorithm for STABLE MULTICUT with running time $f(k, |T|)(n+m)$ for some function f. They posed as open problems, the problem of determining whether there exists an FPT algorithm for STABLE MULTICUT parameterized by k only, and the problem of determining whether there exists an FPT algorithm for STABLE ODD CYCLE TRANSVERSAL with running time $2^{k^{O(1)}} \cdot (n+m)$. The problem of determining whether there exists an FPT algorithm for STABLE MULTICUT parameterized by k was restated by Pilipczuk at the update meeting on graph separation problems in 2013 [Cygan et al. 2013a].

Subsequently, algorithms for ODD CYCLE TRANSVERSAL with running time $4^k k^{\mathcal{O}(1)} \cdot (n+m)$ were found, independently by Iwata et al. [Iwata et al. 2014] and Ramanujan and Saurabh [Ramanujan and Saurabh 2014]. Replacing the call to the algorithm of Reed et al. [Reed et al. 2004] in the algorithm of Marx et al. [Marx et al. 2013] for STABLE ODD CYCLE TRANSVERSAL by either of the two $4^k \cdot k^{\mathcal{O}(1)} \cdot (n+m)$ time algorithms for ODD CYCLE TRANSVERSAL yields a $2^{2^{k^{O(1)}}} \cdot (n+m)$ time algorithm for STABLE ODD CYCLE TRANSVERSAL. However, obtaining a $2^{k^{O(1)}}(n+m)$ time algorithm still remained an open problem.

Using Lemma 1.1, we directly obtain randomized FPT algorithms for STABLE s-t SEPARATOR, STABLE ODD CYCLE TRANSVERSAL, STABLE MULTICUT and STABLE DIRECTED FEEDBACK VERTEX SET on d-degenerate graphs. It is sufficient to apply Lemma 1.1 to obtain an independent set Y containing the solution S, and then run the algorithms for the non-stable version of the problem where all vertices in $V(G) \setminus Y$ are not allowed to go into the solution. For all of the above-mentioned problems, the existing algorithms can easily be made to work even in the setting where some vertices are not allowed to go into the solution.

[2]Independent sets are sometimes called stable sets in the literature. In this paper we stick to independent sets, except for problem names, which are inherited from Marx et al. [Marx et al. 2013].

Lemma 1.1 only applies to graphs of bounded degeneracy. Even though the class of graphs of bounded degeneracy is quire rich (it includes planar graphs, and more generally all graphs excluding a topological minor) it is natural to ask whether Lemma 1.1 could be generalized to work for all graphs. However, if G consists of k disjoint cliques of size n/k each, the best success probability one can hope for is $(k/n)^k$, which is too low to be useful for FPT algorithms.

At a glance the applicability of Lemma 1.1 seems to be limited to problems on graphs of bounded degeneracy. However, there already exist powerful tools in the literature to reduce certain problems on general input graphs to special classes. For us, the *treewidth reduction* of Marx et al. [Marx et al. 2013] is particularly relevant, since a direct application of their main theorem reduces Stable s-t Separator and Stable Odd Cycle Transversal to the same problems on graphs of bounded treewidth. Since graphs of bounded treewidth have bounded degeneracy, we may now apply our algorithms for bounded degeneracy graphs, obtaining new FPT algorithms for Stable s-t Separator and Stable Odd Cycle Transversal on general graphs. Our algorithms have running time $2^{k^{\mathcal{O}(1)}} \cdot (n+m)$, thus resolving, in the affirmative, one of the open problems of Marx et al. [Marx et al. 2013].

One of the reasons that the parameterized complexity of Stable Multicut parameterized by the solution size was left open by Marx et al. [Marx et al. 2013] was that their treewidth reduction does not apply to multi-terminal cut problems when the number of terminals is unbounded. Our second main contribution is a graph sparsification procedure that works for such multi-terminal cut problems. Given a graph G and a set T of terminal pairs, a multicut S of T in G is called a *minimal* multicut of T in G if no proper subset of S is a multicut of T in G. A vertex set X in G is *vertex-k-connected* (or just *k-connected*) if, for every pair u, v of vertices in X, there are k internally vertex disjoint paths from u to v in G.

Theorem 1. *There exists a polynomial time algorithm that given a graph G, a set $T = \{\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_\ell, t_\ell\}\}$ of terminal pairs and an integer k, returns an induced subgraph $G^\star$ of G and a subset $T^\star$ of T which have the following properties.*

- *every minimal multicut of T in G of size at most k is a minimal multicut of T^* in $G^\star$,*
- *every minimal multicut of T^* in $G^\star$ of size at most k is a minimal multicut of T in G, and*
- *$G^\star$ does not contain a $(k+2)$-connected set of size $\mathcal{O}(64^k \cdot k^2)$.*

We remark that excluding a $(k+2)$-connected set of size $\mathcal{O}(64^k \cdot k^2)$ implies that $G^\star$ excludes a clique of size $\mathcal{O}(64^k \cdot k^2)$ as a *topological minor*. In fact, the property of excluding a large $(k+2)$-connected set puts considerable extra restrictions on the graph, on top of being topological minor free, as there exist planar graphs that contain arbitrarily large $(k+2)$-connected sets. The proof of Theorem 1 uses the *irrelevant vertex* technique of Robertson and Seymour [Robertson and Seymour 1995], however, instead of topological arguments for finding an irrelevant vertex we rely on a careful case distinction based on cut-flow duality together with counting arguments based on *important separators*.

Theorem 1 reduces the Stable Multicut problem on general graphs to graphs excluding a clique of size $2^{\mathcal{O}(k)}$ as a topological minor. Since such graphs have bounded degeneracy [Bollobás and Thomason 1998; Komlós and Szemerédi 1996], our algorithm for Stable Multicut on graphs of bounded degeneracy yields an FPT algorithm for the problem on general graphs, resolving the second open problem posed by Marx et al. [Marx et al. 2013].

We remark that a sparsification for directed graphs similar to Theorem 1 powerful enough to handle Directed Feedback Vertex Set is unlikely, since Stable Directed Feedback Vertex Set on general graphs is known to be W[1]-hard [Misra et al. 2012], while our algorithm works on digraphs where the underlying undirected graph has bounded degeneracy.

The algorithms based on Lemma 1.1 are randomized, however they can be derandomized using a new combinatorial object that we call k-independence covering families, that may be of independent interest. We call an independent set of size at most k a k-*independent set*, and we call a family of independent sets an *independent family*. An independent family $\mathcal{F}$ *covers* all k-independent sets of G, if for every k-independent set X in G there exists an independent set $Y \in \mathcal{F}$ such that $X \subseteq Y$. In this case, we call $\mathcal{F}$ a k-*independence covering family*. An algorithm based on Lemma 1.1 can be made deterministic by first constructing a k-independence covering family $\mathcal{F}$, and then looping over all sets $Y \in \mathcal{F}$ instead of repeatedly drawing Y at random using Lemma 1.1.

Since a graph G contains at most n^k independent sets of size at most k, drawing $\mathcal{O}(\binom{k(d+1)}{k} \cdot kd \cdot \log n)$ sets using Lemma 1.1 and inserting them into $\mathcal{F}$ will result in a k-independence covering family with probability at least $1/2$. Hence, for every d and k, every graph G on n vertices of degeneracy at most d has a k-independence covering family of size at most $\mathcal{O}(\binom{k(d+1)}{k} \cdot kd \cdot \log n)$. By direct applications of existing pseudo-random constructions (of *lopsided universal sets* [Fomin et al. 2016] and *perfect hash families* [Fredman et al. 1984]) we show that given a graph G of degeneracy d and integer k one can construct a k-independence covering family of size not much larger than $\mathcal{O}(\binom{k(d+1)}{k} \cdot kd \cdot \log n)$ in time roughly proportionate to its size.

Additionally, we give an efficient construction of k-independence covering families of size at most $f(k)n$ for *nowhere dense* class of graphs [Nešetřil and de Mendez 2008, 2011], based on low treedepth colorings of such graphs. This construction immediately yields FPT algorithms for the considered problems on nowhere dense classes of graphs.

1.1 Proof sketch for Theorem 1.

Towards the proof of Theorem 1, we describe an algorithm that given G, the set T of terminal pairs, an integer k and a $(k+2)$-connected set W of size at least $64^{k+2} \cdot (k+2)^2$, computes a vertex v that does not appear in any minimal multicut of size at most $k+1$. One can show that such a vertex v is *irrelevant* in the sense that G, T has *exactly* the same family of minimal multicuts of size at most k as the graph $G - v$ with the terminal set $T' = \{\{s_i, t_i\} \in T : v \notin \{s_i, t_i\}\}$. The proof of Theorem 1 then follows by repeatedly removing irrelevant vertices, until $|W| \leqslant 64^{k+2} \cdot (k+2)^2$.

Degree 1 Terminals Assumption. In order to identify an irrelevant vertex it is helpful to assume that every terminal s_i or t_i has degree 1 in G, and that no vertex in G appears in more than one terminal pair. To justify this assumption one can, for every pair $\{s_i, t_i\} \in T$, add $k+2$ new degree 1 vertices $s_i^1, s_i^2, \ldots, s_i^{k+2}$ and make them adjacent to s_i, and $k+2$ new degree 1 vertices $t_i^1, t_i^2, \ldots, t_i^{k+2}$ and make them adjacent to t_i. Call the resulting graph G', and make a terminal pair set T' from T by inserting for every pair $\{s_i, t_i\} \in T$ the set $\{\{s_i^j, t_i^j\} : 1 \leqslant j \leqslant k+2\}$ into T'. It is clear that the set of (minimal) multicuts of T' in G' of size at most $k+1$ is the same as the set of (minimal) multicuts of T in G of size at most $k+1$.

Detecting Irrelevant Vertices. In order to identify an irrelevant vertex we investigate the properties of all vertices $v \in W$ for which there exists a minimal multicut of size at most $k+1$ containing v. We will call such vertices *relevant*. Let $v \in W$ be a relevant vertex and let S be a minimal multicut of size at most $k+1$ containing v. Since W is a $(k+2)$-connected set and $|S| \leqslant k+1$, $W \setminus S$ is contained in some connected component C of $G - S$. Since S is a multicut we also have that S is a *pair cut* for T with respect to W in the following sense: for each terminal pair $\{s_i, t_i\}$ at most one of s_i and t_i can reach $W \setminus S$ in $G - S$. This is true because all vertices of $W \setminus S$ lies in the same connected component of $G - S$. Furthermore $S \setminus \{v\}$ can *not* be a pair cut for T with respect to W, because if it happened to be a pair cut, then we can show that $S \setminus \{v\}$ would also have been a multicut, contradicting the minimality of S. We say that $v \in W$ is *essential* if there exists some pair cut S for T with respect to W such that $|S| \leqslant k+1$, $v \in S$ and $S \setminus \{v\}$ is not a pair cut for T with respect to W. The above argument shows that every relevant vertex is essential, and it remains to find a vertex $v \in W$ which is provably not essential.

The algorithm that searches for a non-essential vertex v crucially exploits *important separators*, defined by Marx [Marx 2006]. Given a graph G and two vertex sets A and B, an A-B-*separator* is a vertex set $S \subseteq V(G)$ such that there is no path from $A \setminus S$ to $B \setminus S$ in $G - S$. An A-B-separator S is called a *minimal* A-B-*separator* if no proper subset of S is also an A-B-separator. Given a vertex set S, we define the *reach of* A *in* $G - S$ as the set $R_G(A, S)$ of vertices reachable from A by a path in $G - S$. We can now define a partial order on the set of minimal A-B separators as follows. Given two minimal A-B separators S_1 and S_2, we say that S_1 is "at least as good as" S_2 if $|S_1| \leqslant |S_2|$ and $R_G(A, S_2) \subset R_G(A, S_1)$. In plain words, S_1 "costs less" than S_2 in terms of the number of vertices deleted and S_1 "is pushed further towards B" than S_2 is. A minimal A-B-separator S is an *important* A-B-*separator* if no minimal A-B-separator other than S is at least as good as S. A key insight behind many parameterized algorithms [Chen et al. 2008; Chitnis et al. 2015, 2013; Cygan et al. 2013b; Kratsch et al. 2015; Lokshtanov and Marx 2013; Lokshtanov and Ramanujan 2012; Lokshtanov et al. 2015, 2016b; Marx and Razgon 2014] is that for every k, the number of important A-B-separators of size at most k is at most 4^k [Chen et al. 2009]. We refer the reader to Marx [Marx 2006] and the textbook by Cygan et al. [Cygan et al. 2015] for a more thorough exposition of important separators.

The algorithm that searches for a non-essential vertex v makes the following case distinction. Either there exists a small T-W-separator Z, or there are many vertex disjoint paths from T to W. Here ,we have abused notation by treating T as a set of vertices in the terminal pairs rather than a set of terminal pairs. In the first case, when there exists a T-W-separator Z of size at most $\zeta = 16^{k+1} \cdot 64(k+2)$, we show that every relevant vertex $v \in W$ is contained in some important z-W-separator of size at most $k+1$, for some $z \in Z$. Since there are at most 4^{k+1} such important separators and we can enumerate them efficiently [Chen et al. 2009], the algorithm simply marks all the vertices in W appearing in such an important separator and outputs one vertex that is not marked.

Many Disjoint Paths. If there are at least $16^{k+1} \cdot 64(k+2)$ vertex disjoint paths from T to W we identify a terminal pair $\{s_i, t_i\}$ such that, for every minimal multicut S of size at most $k+1$ for the instance G with terminal set $T \setminus \{\{s_i, t_i\}\}$, S is also a minimal multicut for G with terminal set T. Such a terminal pair is irrelevant in the sense that removing $\{s_i, t_i\}$ from T does not change the family of minimal multicuts of size at most $k+1$. Thus, if we later identify a vertex $v \in W$ that is irrelevant with the reduced terminal set, then v is also irrelevant with respect to the original terminal set. We will say that a terminal pair that isn't irrelevant is *relevant*.

To identify an irrelevant terminal pair we proceed as follows. Without loss of generality, there are $\zeta/2$ vertex disjoint paths from $A = \{s_1, s_2, \ldots s_{\zeta/2}\}$ to W. Thus, for any set S of at most $k+2$ vertices, all of A except for at most $k+2$ vertices can reach $W \setminus S$ in $G - S$.

Let $B = \{t_1, t_2, \ldots t_{\zeta/2}\}$. We have that for every pair cut S for T with respect to W, at most $k+2$ vertices of $B \setminus S$ are reachable from W in $G - S$.

Consider a pair $\{s_i, t_i\}$ with $s_i \in A$ and $t_i \in B$. If $\{s_i, t_i\}$ is relevant, then there must exist a set S of size at most $k+1$ that is a minimal pair cut for G with terminals $T \setminus \{\{s_i, t_i\}\}$ with respect to W, but is not a pair cut with terminal pair set T. We have that t_i is reachable from W in $G - S$, and that $S \cup \{t_i\}$ is a pair cut for T. Let $\hat{B} \subseteq B$ be the set of vertices in B that are reachable from W in $G - (S \cup \{t_i\})$. From the discussion in the previous paragraph it follows that $|\hat{B}| \leqslant k+2$. Thus, $S \cup \{t_i\} \cup \hat{B}$ is a W-B separator of size at most $2(k+2)$. Pick any minimal W-B separator $\hat{S} \subseteq S \cup \{t_i\} \cup \hat{B}$.

We argue that $t_i \in \hat{S}$. To that end we show that there exists a path P from W to t_i in $G - (S \cup \hat{B})$. Thus, if $t_i \notin \hat{S}$ then $\hat{S}$ would be a subset of $S \cup \hat{B}$ and P would be a path from W to B in $G - \hat{S}$, contradicting that $\hat{S}$ is a W-B-separator. We know that there exists a path P from W to t_i in $G - S$ and that P does not visit any vertex in $\hat{B}$ on the way to t_i because all vertices in $\hat{B}$ have degree 1. Hence P is disjoint from $\hat{S}$, yielding the desired contradiction. We conclude that $t_i \in \hat{S}$.

With all of this hard work we have, under the assumption that $\{s_i, t_i\}$ is a relevant pair with $t_i \in B$, exhibited a minimal W-B-separator $\hat{S}$ that contains t_i. There must exist some important W-B-separator $S^\star$ that is at least as good as $\hat{S}$. Since all the vertices of P (except t_i) are reachable from W in $G - \hat{S}$ it follows that $t_i \in S^\star$. We have now shown that if $\{s_i, t_i\}$ is a relevant pair with $t_i \in B$, then there exists a W-B important separator of size at most $2(k+2)$ that contains t_i. The algorithm goes over all W-B important separators of size at most $2(k+2)$ and marks all vertices appearing in such important separators. Since $\zeta/2 > 4^{2(k+2)} \cdot 2(k+2)$ it follows that some vertex t_i in B is left unmarked. The pair (s_i, t_i) is then an irrelevant pair. This concludes the proof sketch that there exists a polynomial time algorithm that given G, T, k and W finds an irrelevant vertex in W, provided that W is large enough.

Finding a Large $(k+2)$-Connected Set. We have shown how to identify an irrelevant vertex given a $(k+2)$-connected set W of large size. But how to find such a set W, if it exists? Given G we can in polynomial time build an auxiliary graph G^* that has the same vertex set as G. Two vertices in G^* are adjacent if there are at least $k+2$ internally vertex disjoint paths between them in G. Clearly $k+2$-connected sets in G are cliques in G^* and vice versa. However, finding cliques in general graphs is $W[1]$-hard, and is believed to have an approximation even in FPT time. To get around this obstacle we exploit the special structure of G^*.

A $(k+2)$-connected set W in G of size at least $64^{k+2} \cdot 4(k+2)^2$ induces a subgraph of G^* where every vertex has degree at least $(k+2)$. Thus the degeneracy of G^* is at least $64^{k+2} \cdot 4(k+2)^2$. A modification of a classic result of Mader [Mader 1972] (see also Diestel [Diestel 2000] and lecture notes of Sudakov [Sudakov 2016]) shows that every graph of degeneracy at least $4d$ contains a $(d+1)$-connected set of size at least $d+2$, and that such a set can be computed in polynomial time. We apply this result with $d = 64^{k+2} \cdot (k+2)^2 - 1$ to obtain a $(64^{k+2} \cdot (k+2)^2)$-connected set in W^* in G^* of size at least $64^{k+2} \cdot (k+2)^2$. A simple argument shows that W^* is also a $(k+2)$-connected set in G. We may now apply the algorithm to detect irrelevant vertices using W^*. This concludes the proof sketch of Theorem 1.

Guide to the paper. In Section 2 we introduce basic notations and some well known results needed for our work. In Section 3 we define independence covering families and give constructions of such families. This allows to derandomize algorithms based on Lemma 1.1. We then construct independence covering families for nowhere dense classes of graphs, and show some barriers to further generalizations of our results. A reader content with randomized FPT algorithms may skip this section altogether. In Section 4 we show the applicability of Lemma 1.1 (or independence covering families) by designing FPT algorithms for STABLE s-t SEPARATOR, STABLE ODD CYCLE TRANSVERSAL, STABLE MULTICUT and STABLE DIRECTED FEEDBACK VERTEX SET on d-degenerate graphs. In Section 5, we explain how the algorithms from Section 4 combined with the treewidth reduction procedure of Marx et al. [Marx et al. 2013] lead to FPT algorithms for some of the considered problems on general graphs. In Section 6 we prove Theorem 1. This is the most technically challenging part of the paper, and may be read independently of the other sections.

2 PRELIMINARIES

We use $\mathbb{N}$ to denote the set of natural numbers starting from 0. For $t \in \mathbb{N}$, $[t]$ is a shorthand for $\{1, \ldots, n\}$. The symbol log denotes natural logarithm and e denotes the base of natural logarithm. For a set U and $t \in \mathbb{N}$, we use 2^U and $\binom{U}{t}$ to denote the power set of U and the set of subsets of U of size t, respectively. For a function $f : D \to R$, $X \subseteq D$ and $Y \subseteq R$, we denote $f(X) = \{f(x) : x \in X\}$ and $f^{-1}(Y) = \{d : f(d) \in Y\}$.

FACT 2.1. $\frac{1}{n}\left[\left(\frac{k}{n}\right)^{-k}\left(\frac{n-k}{n}\right)^{-(n-k)}\right] \leqslant \binom{n}{k} \leqslant \left[\left(\frac{k}{n}\right)^{-k}\left(\frac{n-k}{n}\right)^{-(n-k)}\right].$

Graphs. Throughout our presentation, given a (di)graph G, n denotes the number of vertices in G and m denotes the number of (arcs)edges in G. We use the term *graphs* to represent undirected graphs. For a (di)graph G, $V(G)$ denotes its vertex set, $A(G)$ denotes arc set in case of digraphs, and $E(G)$ denotes edge set in case of graphs. For any positive integers a, b, we denote by $K_{a,b}$ the complete bipartite graph with a vertices in one part and b vertices in the other part. Let G be a (di)graph. For any $X \subseteq V(G)$, $G[X]$ denotes the induced graph on the vertex set X. By $G - X$, we denote the (di)graph $G[V(G) \setminus X]$. When $X = \{v\}$, we use $G - v$ to denote the graph $G - \{v\}$. For a set $Y \subseteq E(G)$, $G - Y$ denotes the (di)graph obtained from G by deleting the edges in Y. For any $u, v \in V(G)$, $d_G(u, v)$ denotes the number of (arcs)edges on the shortest path from u to v in G. For a graph G, for any $u, v \in V(G)$, uv denotes the edge with endpoints u to v. For any $v \in V(G)$, $N_G(v)$ denotes the neighbors of v in G, that is, $N_G(v) = \{u : uv \in E(G)\}$. The degree of a vertex v in G, denoted by $\mathsf{deg}_G(v)$, is equal to the number of neighbors of v in G, that is, $\mathsf{deg}_G(v) = |N_G(v)|$. The minimum degree of G is the minimum over the degrees of all its vertices. If D is a digraph, then for any $u, v \in V(D)$, uv denotes the arc from u to v. By $\overleftarrow{D}$, we denote the digraph obtained from D by reversing each of its arcs. For any $v \in V(D)$, $N^+_D(v)$ denotes the out-neighbors of v in D and $N^-_D(v)$ denotes the in-neighbors of v in D, that is, $N^+_D(v) = \{u : vu \in A(D)\}$ and $N^-_D(v) = \{u : uv \in A(D)\}$. For any $X \subset V(D)$, $N^+_D(X) = \{u : u \in V(D) \setminus X$ and there exists $v \in X$ such that $vu \in A(D)\}$ and $N^-_D(X) = \{u : u \in V(D) \setminus X$ and there exists $v \in X$ such that $uv \in A(D)\}$. For a graph G, $\mathbf{tw}(G)$ denotes the treewidth of G.

For a non-negative integer d, a graph G is called a d-*degenerate graph* if for every subgraph H of G there exists $v \in H$ such that $\mathsf{deg}_H(v) \leqslant d$. The degeneracy of a graph G, denoted by $\mathsf{degeneracy}(G)$, is the least integer d, for which G is d-degenerate. If there exists a subgraph H of G such that the minimum degree of H is at least d, we say that the degeneracy of G is at least d. For a d-degenerate graph G, a d-*degeneracy sequence* of G is an ordering of the vertices of G, say $\sigma : V(G) \to [|V(G)|]$, such that σ is a bijection and, for any $v \in V(G)$,

$|N_G(v) \cap \{u : \sigma(u) > \sigma(v)\}| \leqslant d$. For a given degeneracy sequence σ and a vertex $v \in V(G)$, the vertices in $N_G(v) \cap \{u : \sigma(u) > \sigma(v)\}$ are called the *forward neighbors* of v in σ, and this set of forward neighbors is denoted by $N^f_{G,\sigma}(v)$. The following proposition says we can find d-degeneracy sequence of a graph in linear time.

Proposition 2.1 ([Matula and Beck 1983]). *If G is a d-degenerate graph, for some non-negative integer d, then a d-degeneracy sequence of G exists and can be found in time $\mathcal{O}(n+m)$.*

Graphs Separators. For (di)graph G, $X, Y \subseteq V(G)$, an X-Y-*separator* in G is a subset $C \subseteq V(G)$, such that there is no path from a vertex in $X \setminus C$ to a vertex in $Y \setminus C$ in $G - C$. For (di)graph G, $s, t \in V(G)$ an s-t-*separator* in G is a subset $C \subseteq V(G) \setminus \{s, t\}$ such that there is no path from s to t in $G - C$. Similarly, for $X, Y \subseteq V(G)$, a (X, Y) separator is a subset $C \subseteq V(G)$ such that there is no path from any vertex of X to any vertex of Y in $G - C$. The size of a separator is equal to the cardinality of the separator. A *minimum* s-t-*separator* in G is the one with the minimum number of vertices. A set $Y \subseteq V(G)$ is a *min vertex cut (mincut)* of G if $G - Y$ has at least two components.

Since, checking whether there is an s-t-separator of weight at most k (here a non-negative integer weight function on $V(G)$ is given) can be done by running at most k rounds of the classical Ford-Fullkerson algorithm, Proposition 2.2 follows.

Proposition 2.2. *Given a (di)graph G, $s, t \in V(G)$, an integer k and $w : V(G) \to \mathbb{N}$, an s-t-separator of weight at most k, if it exists, can be found in time $\mathcal{O}(k \cdot (n+m))$. Also, a minimum s-t-separator can be found in time $\mathcal{O}(mn)$.*

Proposition 2.3 ([Stoer and Wagner 1997]). *A mincut of a (di)graph G can be found in time $\mathcal{O}(m + n \log n)$.*

3 TOOL I: INDEPENDENCE COVERING LEMMA

In this section we give constructions of k-independence covering families, which are useful in derandomizing algorithms based on Lemma 1.1. Towards this we first formally define the notion of k-independence covering family – a family of independent sets of a graph G which covers all independent sets in G of size at most k.

Definition 3.1 (k-Independence Covering Family). *For a graph G and $k \in \mathbb{N}$, a family of independent sets of G is called an independence covering family for (G, k), denoted by $\mathcal{F}(G, k)$, if for any independent set X in G of size at most k, there exists $Y \in \mathcal{F}(G, k)$ such that $X \subseteq Y$.*

Observe that for any pair (G, k), there exists an independence covering family of size at most $\binom{n}{k}$ containing all independent sets of size at most k. We show that, if G has bounded degeneracy, then k-independence covering family of "small" size exists. In fact, we give both randomized and deterministic algorithms to construct such a family of "small" size for graphs of bounded degeneracy. In particular, we prove that if G is d-degenerate, then one can construct an independent set covering family for (G, k) of size $f(k, d) \cdot \log n$, where f is a function depending only on k and d. We first give the randomized algorithm for constructing k-independence covering family. Towards this we use the algorithm described in Lemma 1.1. For an ease of reference we present the algorithm given in Lemma 1.1 here.

Lemma 3.1 (Randomized Independence Covering Lemma). *There is an algorithm that given a d-degenerate graph G and $k \in \mathbb{N}$, outputs a family $\mathcal{F}(G, k)$ such that* (a)

Algorithm 1: Input is (G, k), where G is a d-degenerate graph and $k \in \mathbb{N}$

Construct a d-degeneracy sequence σ of G, using Proposition 2.1.
Set $p = \frac{1}{d+1}$. Independently color each vertex $v \in V(G)$ black with probability p and white with probability $(1 - p)$.
Let B and W be the set of vertices colored black and white, respectively.
$Z := \{v \in B \mid N^f_{G,\sigma}(v) \cap B = \emptyset\}$.
return Z

$\mathcal{F}(G, k)$ *is an independence covering family for* (G, k) *with probability at least* $1 - \frac{1}{n}$, (b) $|\mathcal{F}(G, k)| \leqslant \binom{k(d+1)}{k} \cdot 2k^2(d + 1) \cdot \log n$, *and* (c) *the running time of the algorithm is* $\mathcal{O}(|\mathcal{F}(G, k)| \cdot (n + m))$.

Proof. Let $t = \binom{k(d+1)}{k} \cdot k(d+1)$. We now explain the algorithm to construct the family $\mathcal{F}(G, k)$ mentioned in the lemma. We run Algorithm 1 (Lemma 1.1) $\gamma = t \cdot 2k \log n$ times. Let $Z_1, \ldots, Z_\gamma$ be the sets that are output at the end of each iteration of Algorithm 1. Let $\mathcal{F}(G, k)$ be the collection of distinct $Z_i{}'$s. Clearly, $|\mathcal{F}(G, k)| \leqslant t \cdot 2k \log n = \binom{k(d+1)}{k} \cdot 2k^2(d+1) \cdot \log n$. Thus condition (b) is proved. The running time of the algorithm (condition (c)) follows from Lemma 1.1.

Now we prove condition (a) of the lemma. Fix an independent set X in G of cardinality at most k. By Lemma 1.1, we know that for any $Z \in \mathcal{F}(G, k)$, $\Pr[X \subseteq Z] \geqslant \frac{1}{t}$. Thus the probability that there does not exist a set $Z \in \mathcal{F}(G, k)$ such that $X \subseteq Z$ is at most $(1 - \frac{1}{t})^{|\mathcal{F}(G,k)|} \leqslant e^{-2k \log n} = n^{-2k}$. The last inequality follows from a well-known fact that $(1 - a) \leqslant e^{-a}$ for any $a \geqslant 0$. Since the total number of independent sets of size at most k in G is upper bounded by n^k, by the union bound, the probability that there exists an independent set of size at most k which is not a subset of any set in $\mathcal{F}(G, k)$ is upper bounded by $n^{-2k} \cdot n^k = n^{-k} \leqslant 1/n$. This implies that $\mathcal{F}(G, k)$ is an independence covering family for (G, k) with probability at least $1 - \frac{1}{n}$. □

Together with Fact 2.1, Lemma 3.1 implies the following corollary.

Corollary 1. *The cardinality of the family* $\mathcal{F}(G, k)$ *constructed by the algorithm of Lemma 3.1 is at most* $(e(d + 1))^k \cdot 2k^2(d + 1) \cdot \log n$.

Proof. From Lemma 3.1,

$$\begin{aligned}
|\mathcal{F}(G, k)| &\leqslant \binom{k(d+1)}{k} \cdot 2k^2(d+1) \cdot \log n \\
&\leqslant (d+1)^k \cdot \left(\frac{1+d}{d}\right)^{kd} \cdot 2k^2(d+1) \cdot \log n && \text{(By Fact 2.1)} \\
&= \left[(d+1)\left(1 + \frac{1}{d}\right)^d\right]^k \cdot 2k^2(d+1) \cdot \log n \\
&\leqslant (e(d+1))^k \cdot 2k^2(d+1) \cdot \log n && \left(\text{Because } \left(1 + \frac{1}{d}\right)^d \leqslant e, \forall d > 0\right).
\end{aligned}$$

This completes the proof. □

Deterministic Constructions. The two deterministic algorithms, that we give, are obtained from the randomized algorithm presented in Lemma 3.1 by using the n-p-q-lopsided-universal family [Fomin et al. 2016] and the (n, ℓ, ℓ^2)-perfect hash family [Fredman et al. 1984], respectively. Both of our deterministic constructions basically replace the random coloring of the vertices in Line 2 of Algorithm 1 by a coloring defined by n-p-q-lopsided universal family and the (n, ℓ, ℓ^2)-perfect hash family, respectively. In the following, we first define the n-p-q-lopsided-universal family, state Proposition 3.1 (an algorithm to construct an n-p-q-lopsided family of "small" size) which is followed by our first deterministic algorithm (Lemma 3.2).

Definition 3.2 (n-p-q-lopsided-universal family [Fomin et al. 2016]). *A family $\mathcal{F}$ of sets over a universe U is an n-p-q-lopsided-universal family if for every $A \in \binom{U}{p}$ and $B \in \binom{U \setminus A}{q}$, there is an $F \in \mathcal{F}$ such that $A \subseteq F$ and $B \cap F = \emptyset$.*

Proposition 3.1 (Lemma 4.2, [Fomin et al. 2016]). *There is an algorithm that given $n, p, q \in \mathbb{N}$ and a universe U, runs in time $\mathcal{O}(\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot n \log n)$, and outputs an n-p-q-lopsided universal family $\mathcal{F}$ of size at most $\binom{p+q}{p} \cdot 2^{o(p+q)} \cdot \log n$.*

Lemma 3.2 (Deterministic Independence Covering Lemma). *There is an algorithm that given a d-degenerate graph G and $k \in \mathbb{N}$, runs in time $\mathcal{O}(\binom{k(d+1)}{k} \cdot 2^{o(k(d+1))} \cdot (n+m) \log n)$, and outputs a k-independence covering family for (G, k) of size at most $\binom{k(d+1)}{k} \cdot 2^{o(k(d+1))} \cdot \log n$.*

Proof. Let $\mathcal{S}$ be the n-p-q-lopsided-universal family constructed using Proposition 3.1 for $n = |V(G)|$, $p = k$ and $q = kd$. For each $S \in \mathcal{S}$, we run Algorithm 1, where Line 2 is replaced as follows: for each vertex in S, we color it black and we color all the other vertices white. More precisely, we run Algorithm 1 for each $S \in \mathcal{S}$, replacing Line 2 by the procedure just defined, and output the collection $\mathcal{F}(G, k)$ of sets returned at the end of each iteration. The size bound on $|\mathcal{F}(G, k)|$ follows from Proposition 3.1 and the running time of the algorithm follows from the fact that each run of Algorithm 1 takes $\mathcal{O}(n + m)$ time.

We now show that $\mathcal{F}(G, k)$ is, indeed, an independent set covering family for (G, k). Let X be an independent set of cardinality at most k in G. Let σ be the d-degenerate sequence constructed in Line 1 of Algorithm 1. Let $Y = \cup_{v \in X} N^f_{G,\sigma}(v)$. Since X is independent, $X \cap Y = \emptyset$. Furthermore, since σ is a d-degeneracy sequence and $|X| \leqslant k$, we have that $|Y| \leqslant kd$. By the definition of n-p-q-lopsided-universal family, there is a set $S \in \mathcal{S}$ such that $X \subseteq S$ and $S \cap Y = \emptyset$. Consider the run of Algorithm 1 for the set S. In this run, we have that $X \subseteq B$ and $Y \subseteq W$. From the definition of X, Y and Z (set constructed in Line 4), we have that $X \subseteq Z$. This implies that $\mathcal{F}(G, k)$ is an independence covering family of (G, k). This completes the proof. □

Note that when the algorithm of Lemma 3.2 gets as input a graph G whose degeneracy is polynomial in k, the algorithm of Lemma 3.2 returns an independence covering family $\mathcal{F}$ of size that is single-exponential in k. If the degeneracy of the input graph is exponential in k, then the algorithm of Lemma 3.2 can not guarantee a single-exponential sized family. In such cases, our next deterministic algorithm gives better guarantees. The next deterministic algorithm for computing "small" sized independence covering family for graphs of bounded degeneracy uses the notion of (n, ℓ, ℓ^2)-perfect hash family. The algorithm is described in Lemma 3.3.

Definition 3.3 ((n, ℓ, q)-perfect hash family). *For non-negative integers n and q, a family of functions $f_1, \ldots, f_t$ from a universe U of size n to a universe of size q is called a (n, ℓ, q)-perfect hash family, if for any $S \subseteq U$ of size at most ℓ, there exists $i \in [t]$, such that f_i is injective on S.*

Proposition 3.2 ([Fredman et al. 1984]). *For any non-negative integers n and ℓ, and any universe U on n elements, a (n, ℓ, ℓ^2)-perfect hash family of size $\ell^{\mathcal{O}(1)} \cdot \log n$ can be computed in time $\ell^{\mathcal{O}(1)} \cdot n \log n$.*

Lemma 3.3. *There is an algorithm that given a d-degenerate graph G and $k \in \mathbb{N}$, runs in time $\mathcal{O}(\binom{k^2(d+1)^2}{k} \cdot (k(d+1))^{\mathcal{O}(1)} \cdot (n+m)\log n)$, and outputs an independence covering family for (G, k) of size at most $\binom{k^2(d+1)^2}{k} \cdot (k(d+1))^{\mathcal{O}(1)} \cdot \log n$.*

Proof. Let $\ell = k(d+1)$. Let $\mathcal{S}$ be the (n, ℓ, ℓ^2)-perfect hash family constructed by the algorithm of Proposition 3.2. The algorithm replaces the random coloring of vertices in Line 2 of Algorithm 1, by colorings defined by the (n, ℓ, ℓ^2)-perfect hash family, $\mathcal{S}$. In particular, for each $f \in \mathcal{S}$ and $A \in \binom{[\ell^2]}{k}$, we run Algorithm 1, where Line 2 is replaced as follows: all vertices in $f^{-1}(A)$ are colored black and the remaining vertices are colored white. More precisely, we run Algorithm 1 for each $f \in \mathcal{S}$ and $A \in \binom{[\ell^2]}{k}$, replacing Line 2 by the procedure just defined, and output the collection $\mathcal{F}(G, k)$ of sets returned at the end of each iteration. Clearly, the size of the family constructed at the end of this procedure, is $|\mathcal{S}| \cdot \binom{\ell^2}{k}$. The total running time for this algorithm is $\mathcal{O}(|\mathcal{F}(G, k)| \cdot (n+m))$. Thus, the total running time of this algorithm and the size of the output family as claimed in the lemma follows from Proposition 3.2.

We now show that $\mathcal{F}(G, k)$ is, indeed, an independence covering family for (G, k). Let X be an independent set of cardinality at most k in G. Let σ be the d-degeneracy sequence constructed in Line 1 of Algorithm 1. Let $Y = \cup_{v \in X} N^f_{G,\sigma}(v)$. Since X is independent, $X \cap Y = \emptyset$. Since σ is a d-degeneracy sequence and $|X| \leqslant k$, we have that $|Y| \leqslant kd$. Let P and Q be sets of vertices such that, $X \subseteq P, Y \subseteq Q, |P| = k, |Q| = kd$ and $P \cap Q = \emptyset$. By the definition of (n, ℓ, ℓ^2)-perfect hash family, there exists a function, say f, in $\mathcal{S}$, such that f is injective on $P \cup Q$. Now consider the iteration of the algorithm corresponding to f and $f(P)$. In this iteration, the algorithm colors all vertices of P with black and all the remaining vertices with white. Since $X \subseteq P$ and $P \cap Q = \emptyset$, the algorithm colors all vertices of X with black color and all vertices of Y with white color. From the definitions of X, Y and Z (set constructed in Line 4), we have that $X \subseteq Z$. This implies that $\mathcal{F}(G, k)$ is an independence covering family of (G, k). This concludes the proof. ◻

Together with Fact 2.1, Lemma 3.3 implies the following corollary.

Corollary 2. *The size of the family $\mathcal{F}(G, k)$ constructed by the algorithm of Lemma 3.3 is at most $k^k \cdot (d+1)^{2k} \cdot e^{k - \frac{1}{(d+1)^2}} \cdot (k(d+1))^{\mathcal{O}(1)} \cdot \log n$.*

PROOF. Let $\ell = k(d+1)$. From Lemma 3.3,

$$
\begin{aligned}
|\mathcal{F}(G,k)| &\leqslant \binom{\ell^2}{k} \cdot \ell^{\mathcal{O}(1)} \cdot \log n \\
&\leqslant \left(\frac{\ell^2}{k}\right)^k \cdot \left(\frac{\ell^2}{\ell^2 - k}\right)^{\ell^2 - k} \cdot \ell^{\mathcal{O}(1)} \cdot \log n && \text{(By Fact 2.1)} \\
&= k^k \cdot (d+1)^{2k} \cdot \left(1 - \frac{k}{\ell^2}\right)^{k - \ell^2} \cdot \ell^{\mathcal{O}(1)} \cdot \log n \\
&\leqslant k^k \cdot (d+1)^{2k} \cdot e^{k - \frac{k^2}{\ell^2}} \cdot \ell^{\mathcal{O}(1)} \cdot \log n && \text{(Because } (1+p) \leqslant e^p, \forall p > 0) \\
&= k^k \cdot (d+1)^{2k} \cdot e^{k - \frac{1}{(d+1)^2}} \cdot (k(d+1))^{\mathcal{O}(1)} \cdot \log n && \text{(Because } \ell = k(d+1))
\end{aligned}
$$

This completes the proof. □

3.1 Extensions

For some graphs, whose degeneracy is not bounded, it may still be possible to find a "small" sized independence covering family. This is captured by the Corollary 3.

COROLLARY 3. *Let* $d, k \in \mathbb{N}$ *and* G *be a graph. Let* $S \subseteq V(G)$ *be such that* $G - S$ *is* d*-degenerate. There are two deterministic algorithms which given* $d, k \in \mathbb{N}$*,* G *and* S*, run in time* $\mathcal{O}(2^{|S|} \cdot \binom{k(1+d)}{k} \cdot 2^{o(k(1+d))} \cdot (n+m) \log n)$ *and* $\mathcal{O}(2^{|S|} \cdot \binom{k^2(1+d)^2}{k} \cdot (k(1+d))^{\mathcal{O}(1)} \cdot (n+m) \log n)$*, and outputs an independence covering family for* (G, k) *of size at most* $2^{|S|} \cdot \binom{k(1+d)}{k} \cdot 2^{o(k(1+d))} \cdot \log n$ *and* $2^{|S|} \cdot \binom{k^2(1+d)^2}{k} \cdot (k(1+d))^{\mathcal{O}(1)} \cdot \log n$ *respectively.*

PROOF. Let $G' = G - S$. By the property of S, we know that G' is d-degenerate. We first apply Lemma 3.2 and get a k-independent set covering family $\mathcal{F}'$ for (G', k). Then we output the family

$$\mathcal{F}(G,k) = \{(A \cup B) \setminus N_G(B) \mid A \in \mathcal{F}', B \subseteq S \text{ is an independent set in } G\}.$$

We claim that $\mathcal{F}(G,k)$ is a k-independence covering family for (G,k). Towards that, first we prove that all sets in $\mathcal{F}(G,k)$ are independent sets in G. Let $Y \in \mathcal{F}$. We know that $Y = (A \cup B) \setminus N_G(B)$, for some $A \in \mathcal{F}'$ and $B \subseteq S$which is an independent set in G. By the definition of $\mathcal{F}'$, A is an independent set in G. Since A and B are independent sets in G, $Y = (A \cup B) \setminus N_G(B)$ is an independent set in G. Now we show that for any independent set X in G of cardinality at most k, there is an independent set containing X in $\mathcal{F}(G,k)$. Let $X = X' \uplus X''$, where $X' = X \setminus S$ and $X'' = X \cap S$. By the definition of $\mathcal{F}'$, there is a set $Z \in \mathcal{F}'$ such that $X' \subseteq Z$. Then the set $(Z \cup X'') \setminus N_G(X'') \in \mathcal{F}(G,k)$ is the required independent set containing X. Observe that $|\mathcal{F}(G,k)| \leqslant |\mathcal{F}'| \cdot 2^{|S|}$. Also, the running time of this algorithm is equal to the time taken to compute $\mathcal{F}'$ plus $|\mathcal{F}(G,k)| \cdot (n+m)$. Thus, the running time and the bound on the cardinality of $\mathcal{F}(G,k)$ as claimed in the lemma follows from Lemmas 3.2 and 3.3. □

3.2 Nowhere Dense Graphs

In this section, we give an efficient construction of k-independence covering families of size at most $f(k) \cdot n$ for every *nowhere dense* class of graphs [Nešetřil and de Mendez 2008, 2011], based on low treedepth colorings of such graphs. The class of nowhere dense graphs is a common generalization of proper minor closed classes, classes of graphs with bounded degree, graph class locally excluding a fixed graph H as minor and classes of bounded expansion (see [Nešetřil and de Mendez 2011, Figure 3]).

In order to define the class of nowhere dense graphs, we need several new definitions.

Definition 3.4 (Shallow minor). *A graph M is an* r-shallow minor *of G, where r is an integer, if there exists a set of disjoint subsets $V_1, \dots, V_{|M|}$ of $V(G)$ such that*

(1) each graph $G[V_i]$ is connected and has radius at most r, and
(2) there is a bijection $\psi\colon V(M) \to \{V_1, \dots, V_{|M|}\}$ such that for every edge $uv \in E(M)$ there is an edge in G with one endpoint in $\psi(u)$ and second in $\psi(v)$.

The set of all r-shallow minors of a graph G is denoted by $G \triangledown r$. Similarly, the set of all r-shallow minors of all the members of a graph class $\mathcal{G}$ is denoted by $\mathcal{G} \triangledown r = \cup_{G \in \mathcal{G}} (G \triangledown r)$.

We first introduce the definition of a nowhere dense graph class; let $\omega(G)$ denotes the size of the largest clique in G and $\omega(\mathcal{G}) = \sup_{G \in \mathcal{G}} \omega(G)$.

Definition 3.5 (Nowhere dense). *A graph class $\mathcal{G}$ is* nowhere dense *if there exists a function $f_\omega\colon \mathbb{N} \to \mathbb{N}$ such that for all r we have that $\omega(\mathcal{G} \triangledown r) \leqslant f_\omega(r)$.*

We will mostly rely on the low treedepth colorings of nowhere dense graph classes. Towards that, we first define the notion of treedepth.

Definition 3.6 (Treedepth of a graph). *A* treedepth decomposition *of a graph G is a rooted forest F on the vertex set $V(G)$, that is $V(F) = V(G)$, such that for every edge $uv \in E(G)$, the endpoints u and v are in ancestor-descendant relation. The* height *of a rooted forest F, denoted by $\mathbf{height}(F)$, is the maximum number of vertices on a simple path from the root of F to a leaf in F. The* treedepth *of G, denoted $\mathbf{td}(G)$, is the least $d \in \mathbb{N}$ such that there exists a treedepth decomposition F of G with $\mathbf{height}(F) = d$.*

Proposition 3.3. *Let G be a graph. Then,* $\mathsf{degeneracy}(G) \leqslant \mathbf{tw}(G) \leqslant \mathbf{td}(G) - 1$.

Next we define the notion of *treedepth colorings* and state a result that shows that nowhere dense graph classes admit low treedepth colorings.

Definition 3.7 (Treedepth coloring). *An r-treedepth coloring of G is a coloring such that any $r' \leqslant r$ color classes induce a subgraph with treedepth at most $r' - 1$. The minimum number of colors of such a coloring of G is denoted by $\mathbf{tdcolr}_r(G)$.*

Proposition 3.4 (Theorem 5.6,[Nešetřil and de Mendez 2011]). *Let $\mathcal{G}$ be a nowhere dense graph class. Then there is a function $f(r, \varepsilon)$ such that $\mathbf{tdcolr}_r(G) \leqslant f(r, \varepsilon) \cdot n^\varepsilon$ for every integer $r \geqslant 0$, $G \in \mathcal{G}$, and real $\varepsilon > 0$. Furthermore, a $f(r, \varepsilon) \cdot n^\varepsilon$-treedepth coloring of G can be obtained in time $\mathcal{O}(f(r, \varepsilon) \cdot n^{1+o(1)})$.*

We refer readers to the book by Nesetril and Ossona de Mendez [Nesetril and de Mendez 2012] for a detailed exposure to nowhere dense classes of graphs, their alternate characterization and several properties about it. See also [Grohe et al. 2013].

We present two algorithms for the construction of "small" sized independence covering family for (G, k), where G belongs to the class of nowhere dense graphs. There are two core ingredients of these algorithms - the first one are the deterministic algorithms of Lemmas 3.2 and 3.3 that compute an independence covering family for bounded degeneracy graphs, and the second one is Proposition 3.4 that states that for any integer r, the vertices of the graph belonging to the nowhere dense graph class, can be partitioned in such a way that the graph induced on any of the $i \leqslant r$ parts of this partition has treedepth bounded by i. Since bounded treedepth implies bounded degeneracy [Nesetril and de Mendez 2012, Propositon 6.4], we can compute independence covering family for each such subgraph (that has bounded treedepth) using the algorithms of Lemmas 3.2 and 3.3. We can then combine these independent set covering families of such subgraphs to obtain an independence covering family for the whole graph. This whole idea is formalized in Lemma 3.4.

LEMMA 3.4. *Let* G *be a graph such that* $G \in \mathcal{G}$*, where* $\mathcal{G}$ *is a class of nowhere dense graphs. For any* $k \in \mathbb{N}$*, and any* $\mathsf{delta} \in \mathbb{R}$ *such that* $\delta > 0$*, there is a deterministic algorithm that runs in time*

$$\mathcal{O}\left(f(k, \frac{\delta}{k}) \cdot n^{1+o(1)} + g(k) \cdot 2^{\mathcal{O}(k \log k)} \cdot n^{\delta}(n+m) \log n\right)$$

, and outputs a k*-independence covering family for* (G, k) *of size* $\mathcal{O}(g(k) \cdot 2^{\mathcal{O}(k \log k)} \cdot n^{\delta} \log n)$*, where* f *is a function defined in Proposition 3.4 and* $g(k) = (f(k, \frac{\delta}{k}))^k$*.*

PROOF. For the given graph G, and an integer k, set $\epsilon = \frac{\delta}{k}$. Now compute the partition of $V(G)$ into $p = \mathcal{O}(f(k, \frac{\delta}{k}) \cdot n^{\delta/k})$ parts, say $V_1, \ldots, V_p$, using the algorithm of Proposition 3.4. For all $A \in \binom{[p]}{k}$, let $G_A = G[\cup_{i \in A} V_i]$. Let $\mathcal{F}_A$ be an independence covering family for (G_A, k). Let $\mathcal{F}(G, k) = \cup_{A \in \binom{[p]}{k}} \mathcal{F}_A$. We claim that $\mathcal{F}(G, k)$ is a k-independence covering family for (G, k). Let X be an independent set in G of size at most k. We show that there exists a set $Y \in \mathcal{F}(G, k)$ such that $X \subseteq Y$. Let $A' \subseteq [p]$ such that for any $i \in [p]$, $X \cap V_i \neq \emptyset$ if and only if $i \in A'$. Let $A \subseteq [p]$ such that $A' \subseteq A$ and $|A| = k$. Observe that X is an independent set in G_A and then thee exists $Y \in \mathcal{F}_A$ such that $X \subseteq Y$. Since $\mathcal{F}(G, k) = \cup_{\binom{[p]}{k}} \mathcal{F}_A$, $Y \in \mathcal{F}(G, k)$.

Observe that, to compute $\mathcal{F}(G, k)$, one needs to compute the partition $V_1, \ldots, V_p$ and $\mathcal{F}_A$ for each $A \in \binom{[p]}{k}$. From Proposition 3.4, $\mathbf{td}(G_A) \leqslant k-1$ and thus, the degeneracy of $G_A \leqslant k-1$ ([Nesetril and de Mendez 2012, Proposition 6.4]). Thus, from Proposition 3.4 and, Lemmas 3.3, there is a deterministic algorithm that runs in time

$$\mathcal{O}\left(f(k, \frac{\delta}{k}) \cdot n^{1+o(1)} + \binom{p}{k} \cdot 2^{\mathcal{O}(k \log k)} \cdot (n+m) \log n\right),$$

and outputs an independence covering family for (G, k) of size $\mathcal{O}(\binom{p}{k} \cdot 2^{\mathcal{O}(k \log k)} \cdot \log n)$. Here, f is the function described in Proposition 3.4. Observe that $\binom{p}{k} \leqslant (f(k, \frac{\delta}{k}) n^{\delta/k})^k \leqslant g(k) \cdot n^{\delta}$, where $g(k) = (f(k, \frac{\delta}{k}))^k$. This concludes the proof. □

3.3 Barriers

In this subsection we show that we can not get small independence covering families on general graphs. We also show that we can not get small covering families when we generalize the notion of "independent set" to something similar even on graphs of bounded degeneracy.

Independence covering family for general graphs. Let k be a positive integer. Consider the graph G on n vertices, where n is divisible by k, which is a disjoint collection of k cliques on $\frac{n}{k}$ vertices each. Let $C_1, \ldots, C_k$ be the disjoint cliques that comprise G. Let $\mathcal{F}(G, k)$ be a k-independence covering family for (G, k). Then, we claim that, $|\mathcal{F}(G, k)| \geqslant \left(\frac{n}{k}\right)^k$. Consider the family $\mathcal{I}$ of independent sets of G of size at most k defined as $\mathcal{I} = \{\{v_1, \ldots, v_k\} : \forall i \in [k], v_i \in C_i\}$. Note that $|\mathcal{I}| = \left(\frac{n}{k}\right)^k$. We now prove that, it is not the case that there exists $Y \in \mathcal{F}(G, k)$ such that for two distinct sets $X_1, X_2 \in \mathcal{I}$, $X_1, X_2 \subseteq Y$. This would imply that $|\mathcal{F}(G, k)| \geqslant \left(\frac{n}{k}\right)^k$. Suppose, for the sake of contradiction, that there exists $Y \in \mathcal{F}(G, k)$ and

$X_1, X_2 \in \mathcal{I}$ such that $X_1 \neq X_2, X_1 \subseteq Y$ and $X_2 \subseteq Y$. Since $X_1 \neq X_2$, there exist $u \in X_1$ and $v \in X_2$ such that $u, v \in C_i$ for some $i \in [k]$. Since $X_1 \subseteq Y$ and $X_2 \subseteq Y$, $u, v \in Y$, which contradicts the fact that Y is an independent set in G (because $uv \in E(G)$).

Induced matching covering family for disjoint union of stars. We show that if we generalize independent set to induced matching, then we can not hope for small covering families even on the disjoint union of *star* graphs, which are graphs of degeneracy one.

Definition 3.8 (Induced Matching Covering Family). *For a graph* G *and a positive integer* k*, a family* $\mathcal{M} \subseteq 2^{V(G)}$ *is called an* induced matching covering family *for* (G, k) *if for all* $Y \in \mathcal{M}$*,* $G[Y]$ *is a matching, that is, each vertex of* Y *has degree exactly one in* $G[Y]$*, and for any induced matching* M *in* G *on at most* k *vertices, there exists* $Y \in \mathcal{M}$ *such that* $V(M) \subseteq Y$*.*

Let k be a positive integer. Consider the graph G on n vertices, where $2n$ is divisible by k, which is a disjoint collection of k stars on $\frac{2n}{k}$ vertices ($K_{1,\frac{2n}{k}-1}$). That is each connected component of G is isomorphic to $K_{1,\frac{2n}{k}-1}$. Let $\mathcal{R}$ be the set of all maximal matchings in G. Each matching in R consists of $\frac{k}{2}$ edges, one from each connected component. Observe that all these matchings are induced matchings in G. Union of any two distinct matchings in $\mathcal{R}$ will have a P_3. This implies that the cardinality of any induced matching covering family for (G, k) is at least $|\mathcal{R}| = (\frac{2n}{k} - 1)^{\frac{k}{2}}$.

r*-scattered covering family for disjoint union of stars.* Let G be a graph. For any $r \in \mathbb{N}$, $X \subseteq V(G)$ is called an r-*scattered set* in G, if for any $u, v \in V(G)$, $d_G(u, v) > r$. An independent set in G is a 1-scattered set in G.

Definition 3.9 (r-scattered Covering Family). *For any* $r \in \mathbb{N}$*, for a graph* G *and a positive integer* k*, a family* $\mathcal{S} \subseteq 2^{V(G)}$ *is called a* r-scattered covering family *for* (G, k) *if for all* $Y \in \mathcal{S}$*,* Y *is an* r*-scattered set in* G *and for any* $X \subseteq V(G)$ *of size at most* k *such that* X *is an* r*-scattered set in* G*, there exists* $Y \in \mathcal{S}$ *such that* $X \subseteq Y$*.*

Let k be a positive integer. Consider the graph G on n vertices, where n is divisible by k, which is a disjoint collection of k stars on $\frac{n}{k}$ vertices ($K_{1,\frac{n}{k}-1}$). That is each connected component of G is isomorphic to $K_{1,\frac{n}{k}-1}$. Notice that G is a 1-degenerate graph. Let $C_1, \ldots, C_k$ be the components of G. Define $\mathcal{I} = \{\{v_1, \ldots, v_k\} : \forall i \in [k], v_i \in C_i\}$. Clearly each set in $\mathcal{I}$ is a r-scattered set for any $r \in \mathbb{N}$. Moreover, union of any two distinct sets in $\mathcal{I}$ is not a 2-scattered set. This implies that the cardinality of any r-scattered covering family for (G, k) is at least $|\mathcal{I}| = \left(\frac{n}{k}\right)^k$ for any $r \geqslant 2$.

Acyclic covering family for 2-degenerate graphs. We show that covering families for induced acyclic subgraphs on 2-degenerate graphs will have large cardinality.

Definition 3.10 (Acyclic Set Covering Family). *For a graph* G *and a positive integer* k*, a family* $\mathcal{A} \subseteq 2^{V(G)}$ *is called an* acyclic set covering family *for* (G, k) *if for all* $Y \in \mathcal{M}$*,* $G[Y]$ *is a forest and for any* $X \subseteq V(G)$ *of size at most* k *such that* $G[X]$ *is a forest, there exists* $Y \in \mathcal{A}$ *such that* $X \subseteq Y$*.*

Let k be a positive integer. Consider the graph G on n vertices, where $3n$ is divisible by k, which is a disjoint union of $\frac{k}{3}$ complete bipartite graphs $K_{2,\frac{3n}{k}}$. The degeneracy of G is 2. Without loss of generality assume that $\frac{3n}{k}$ is strictly more than 2. Let $H_1, \ldots, H_{\frac{k}{3}}$ be the connected components of G. Let $H_i = (L_i \uplus R_i, E_i)$, where $|L_i| = 2$. Now consider the family of sets $\mathcal{J} = \{L_1 \cup \ldots \cup L_{\frac{k}{3}} \cup \{v_1, \ldots, v_{\frac{k}{3}}\} \mid v_i \in R_i\}$. Each set in $\mathcal{J}$ induces a collection of induced paths on 3 vertices (P_3). Also, union of any two sets in $\mathcal{J}$ contains a cycle on 4 vertices and hence, not acyclic. This implies that the cardinality of any acyclic set covering family for (G, k) is at least $|\mathcal{J}| = \left(\frac{3n}{k} - 2\right)^{\frac{k}{3}}$.

4 APPLICATIONS I: DEGENERATE GRAPHS

In this section we give FPT algorithms for Stable s-t Separator, Stable Odd Cycle Transversal, Stable Multicut and for Stable Directed Feedback Vertex Set on d-degenerate graphs, by applying Lemmas 1.1, 3.2 and 3.3. All these algorithms, except the one for Stable Directed Feedback Vertex Set, are later used as a subroutine to design FPT algorithms on general graphs.

4.1 Stable s-t-Separator

In this subsection, we study the problem of Stable s-t Separator on graphs of bounded degeneracy. The problem is formally defined below.

Stable s-t Separator (SSTS) **Parameter:** k
Input: A graph G, $s, t \in V(G)$ and $k \in \mathbb{N}$.
Question: Is there an s-t-separator S in G of size at most k such that S is an independent set in G?

In [Marx et al. 2013], the authors showed that SSTS is FPT by giving an algorithm that runs in time, that is roughly, $2^{2^{k^{\mathcal{O}(1)}}} \cdot n \cdot \alpha(n, n)$, where α is the inverse Ackermann function. In this section, we give improved algorithm for SSTS when the input graph is d-degenerate, for some non-negative integer d. In particular, we give a randomized algorithm for SSTS, when the input graph is d-degenerate, that runs in time $\mathcal{O}(\binom{k(1+d)}{k} \cdot k^2(1+d) \cdot (n+m))$ and a deterministic algorithm for the same that runs in time $\min\{\mathcal{O}(\binom{k(1+d)}{k} \cdot 2^{o(k(1+d))} \cdot k \cdot (n+m) \log n), \mathcal{O}(\binom{k^2(1+d)^2}{k} \cdot (k(1+d))^{\mathcal{O}(1)} \cdot (n+m) \log n)\}$.

The core idea behind both the algorithms is that, with an independent set covering family for (G, k) at hand, the independent solution to the problem lies inside one of the set in this family. Thus, instead of looking for an independent s-t-separator separator in a graph, one can look for an s-t-separator separator that is contained inside one of the sets in this family. To shave off the log factor in the randomized algorithm, that we would get if we construct an independent set covering family using the algorithm of Lemma 3.1, we use Algorithm 1 in our algorithm instead of constructing the whole $\mathcal{F}(G, k)$ before hand using multiple rounds of Algorithm 1. We now define the annotated version of the s-t-separator separator problem, which will eventually be the core problem that we would be required to solve, in order to give an algorithm for SSTS.

Annotated s-t Separator (ASTS) **Parameter:** k
Input: A graph G, $s, t \in V(G)$, $Y \subseteq V(G)$ and $k \in \mathbb{N}$.
Question: Is there an s-t-separator S of size at most k in G such that $S \subseteq Y$?

Lemma 4.1. ASTS *can be solved in time* $\mathcal{O}(k \cdot (n+m))$.

Proof Sketch. To prove the lemma, we apply Proposition 2.2 on (G, s, t, w, k), where w is defined as follows: $w(v) = 1$ if $v \in Y$ and $k+1$ otherwise. □

Theorem 2. *There is a randomized algorithm which solves* SSTS *on* d*-degenerate graphs with a worst case running time of* $\mathcal{O}(\binom{k(1+d)}{k} \cdot k^2(1+d) \cdot (n+m))$*. If the input is a* Yes *instance, then the algorithm output* Yes *with probability at least* $1 - 1/e$ *and if it is a* No *instance, then the algorithm always outputs* No.

Proof. Our algorithm runs the following two step procedure $\binom{k(1+d)}{k} \cdot k(1+d)$ many times.

(1) Run Algorithm 1 on (G, k) and let Z be its output.
(2) Run the algorithm of Lemma 4.1 on the instance (G, s, t, k, Z) of ASTS.

Our algorithm will output Yes, if Step 2 returns Yes at least once. Otherwise, our algorithm will output No. We now prove the correctness of our algorithm. Since, in Step 1, the output set Z is always an independent set of G, if the algorithm returns Yes, the input instance is a Yes instance. For the other direction, suppose the input instance is a Yes instance. Let X be a solution to it. Since X is an independent set, from Lemma 1.1, $X \subseteq Z$ with probability at least $p = \frac{1}{\binom{k(d+1)}{k} \cdot (k(d+1))}$. Thus, the probability that in all the executions of Step 1, $X \not\subseteq Z$ is at most $(1-p)^{1/p} \leqslant 1/e$. Therefore, the probability that in at least one execution of Step 1, $X \subseteq Z$, is at least $1 - 1/e$. Now, consider the iteration of the algorithm when $X \subseteq Z$. For this iteration, (G, s, t, k, Z) is a Yes instance of ASTS, and thus, our algorithm will output Yes in this iteration. Therefore, if the input instance is a Yes/ instance, our algorithm will output Yes with probability at least $1 - 1/e$.

The running time of our algorithm follows from Lemmas 1.1 and 4.1. ◻

Theorem 3. *There is a deterministic algorithm which solves* SSTS *on* d*-degenerate graphs in time* $\min\{\mathcal{O}(\binom{k(1+d)}{k} \cdot 2^{o(k(1+d))} \cdot k \cdot (n+m)\log n), \mathcal{O}(\binom{k^2(1+d)^2}{k} \cdot (k(1+d))^{\mathcal{O}(1)} \cdot (n+m)\log n)\}$.

Proof Sketch. Let $\mathcal{F}(G, k)$ be an independent set covering family obtained by the algorithm of Lemma 3.2 or Lemma 3.3. Our algorithm runs Step 2 of the procedure described in the proof of Theorem 2 for all $Z \in \mathcal{F}(G, k)$. It outputs Yes if at least one execution gives a Yes answer. Otherwise, it outputs No. The correctness of the algorithm follows from the definition of independent set covering family and Lemma 4.1. The running time of the algorithm follows from Lemmas 3.2, 3.3 and 4.1. ◻

4.2 Stable Odd Cycle Transversal

In this section, we study the problem of Stable Odd Cycle Transversal (also named as Stable Bipartization in [Marx et al. 2013]) where the input graph has bounded degeneracy. The problem is formally defined below.

Stable Odd Cycle Transversal (SOCT)	**Parameter:** k
Input: A graph G and $k \in \mathbb{N}$.	
Question: Is there a set $X \subseteq V(G)$ of size at most k such that X is an independent set in G and $G - X$ is acyclic?	

In [Marx et al. 2013], the authors showed that SOCT is FPT by giving an algorithm that runs in time $2^{2^{k^{\mathcal{O}(1)}}} \cdot n \cdot \alpha(n, n)$, where α is the inverse Ackermann function. In this section, we give an improved algorithm for SOCT when the input graph is d-degenerate, for some non-negative integer d. In particular, we give a randomized algorithm for SOCT on d-degenerate graphs that runs in time $\mathcal{O}(4^k \cdot \binom{k(1+d)}{k} \cdot k^7(1+d) \cdot (n+m))$ and a

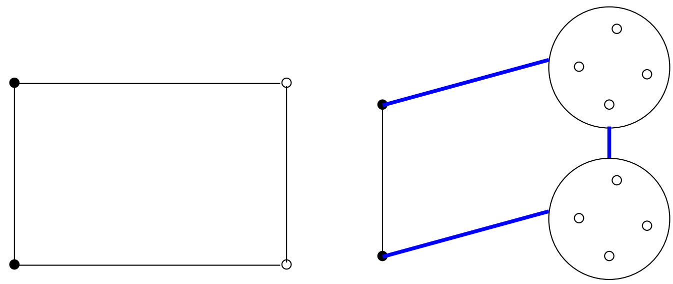


Fig. 1. The graph at the right hand side is obtained by the reduction on the graph at the left hand side, where $k = 3$ and Y is the set of black colored vertices. Thick lines represents all possible edges between two sets of vertices.

deterministic algorithm for the same that runs in time $\min\{\mathcal{O}(4^k \cdot k^6 \cdot \binom{k(1+d)}{k} \cdot 2^{o(k(1+d))} \cdot (n+m)\log n), \mathcal{O}(\binom{4^k \cdot k^2(1+d)^2}{k} \cdot (k(1+d))^{\mathcal{O}(1)} \cdot (n+m)\log n)\}$. As was the case with SSTS, the core idea behind both the algorithms is that, with an independent set covering family for (G, k) at hand, the independent solution to the problem lies inside one of the set in this family. We now define the annotated version of the OCT problem.

Annotated Odd Cycle Transversal (AOCT)	**Parameter:** k
Input: A graph G, $Y \subseteq V(G)$, and $k \in \mathbb{N}$.	
Question: Is there a set $X \subseteq Y$ of size at most k such that $G - X$ is acyclic?	

When $Y = V(G)$ in AOCT, the problem is well-known by the name of Odd Cycle Transversal(OCT). We will need the following result about OCT.

Proposition 4.1 ([Ramanujan and Saurabh 2014]). OCT *can be solved in time* $\mathcal{O}(4^k \cdot k^4 \cdot (n+m))$.

Using Proposition 4.1, we can get the following result about AOCT.

Lemma 4.2. AOCT *can be solved in time* $\mathcal{O}(4^k \cdot k^6 \cdot (n+m))$

Proof sketch. We give a polynomial time reduction from AOCT to OCT as follows. We replace each $v \in V(G) \setminus Y$, with $k+1$ vertices $v_1, \ldots v_{k+1}$ with same neighborhood as v, that is, the neighborhood of $v_1, \ldots v_{k+1}$ are same in the resulting graph (see Figure 4 for an illustration). Let G' be the resulting graph. Then any minimal odd cycle transversal which contain a vertex from $\{v_1, \ldots, v_{k+1}\}$ will also contain all the vertices in $\{v_1, \ldots, v_{k+1}\}$. Thus to find a k sized solution for AOCT, it is enough to find an odd cycle transversal of size k in G'. The total number of vertices in G' is at most $k|V(G)|$ and the total number of edges in G' is at most $(k+1)^2|E(G)|$. Thus the running time of the algorithm follows from Proposition 4.1. $\square$

By applying Lemma 4.2, instead of Lemma 4.1, in Theorems 2 and 3, we get the following theorems.

Theorem 4. *There is a randomized algorithm which solves* SOCT *on* d*-degenerate graphs with a worst case running time of* $\mathcal{O}(4^k \cdot \binom{k(1+d)}{k} \cdot k^7(1+d) \cdot (n+m))$. *If the input is a* Yes *instance, then the algorithm output* Yes *with probability at least* $1 - 1/e$ *and if it is a* No *instance, then the algorithm always outputs* No.

THEOREM 5. *There is a deterministic algorithm which solves* SOCT *on* d*-degenerate graphs in time* $\min\{\mathcal{O}(4^k \cdot k^6 \cdot \binom{k(1+d)}{k} \cdot 2^{o(k(1+d))} \cdot (n+m)\log n), \mathcal{O}(\binom{4^k \cdot k^2 (1+d)^2}{k} \cdot (k(1+d))^{\mathcal{O}(1)} \cdot (n+m)\log n)\}$.

4.3 Stable Directed Feedback Vertex Set

In this section, we study the problem of STABLE DIRECTED FEEDBACK VERTEX SET (SDFVS). SDFVS was shown to be W[1]-HARD in [Misra et al. 2012]. We study SDFVS restricted to the case where the input graph has bounded degeneracy. The problem is formally defined below.

STABLE DIRECTED FEEDBACK VERTEX SET (SDFVS) **Parameter:** k
Input: A digraph D and $k \in \mathbb{N}$.
Question: Is there a set $X \subseteq V(D)$ of size at most k such that S is an independent set in D and $D - S$ is a directed acyclic graph?

As with the algorithms in the previous sections, the algorithm for this problem follows the same outline.

We need to use the known algorithm for DIRECTED FEEDBACK VERTEX SET (DFVS) - the same problem as SDFVS where the solution need not be an independent set.

LEMMA 4.3 ([LOKSHTANOV ET AL. 2016A]). DFVS *can be solved in time* $\mathcal{O}((k+1)! \cdot 4^k \cdot k^5 \cdot (n+m))$.

THEOREM 6. *There is a randomized algorithm which solves* SDFVS *on* d*-degenerate graphs with a worst case running time* $\mathcal{O}((k+1)! \cdot 4^k \cdot \binom{k(1+d)}{k} \cdot k^6(1+d) \cdot (n+m))$. *If the input is a* Yes *instance, then the algorithm outputs* Yes *with probability at least* $1 - 1/e$ *and if it is a* No *instance, then the algorithm always outputs* No.

PROOF. The algorithm runs the following two step procedure $\binom{k(1+d)}{k} \cdot k(1+d)$ many times.

(1) Run Algorithm 1 on (G, k) and let Z be its output.
(2) Construct G' as in Lemma 4.2, that is, add $k+1$ copies for each vertex in $V(G) \setminus Z$ to the graph G such that all of them have the same neighborhood in the resulting graph. Then apply Lemma 4.3 on (G', k).

The proof of correctness of this algorithm is similar in arguments to the proofs of Lemma 4.2 and Theorem 2.

The running time of the algorithm follows from Lemmas 1.1 and 4.3. □

By arguments similar to the proof of Theorem 3, one can prove the following theorem.

THEOREM 7. *There is a deterministic algorithm which solves* SDFVS *on* d*-degenerate graphs in time* $\min\{\mathcal{O}((k+1)! \cdot 4^k \cdot k^5 \cdot \binom{k(1+d)}{k} \cdot 2^{o(k(1+d))} \cdot (n+m)\log n), \mathcal{O}((k+1)! \cdot 4^k \cdot \binom{k^2(1+d)^2}{k} \cdot (k(1+d))^{\mathcal{O}(1)} \cdot (n+m)\log n)\}$.

4.4 Stable Multicut

For a graph G and a set of terminal pairs $T = \{\{s_1, t_1\}, \ldots, \{s_p, t_p\}\}$, $S \subseteq V(G)$ is a *multicut* of T in G if $G - S$ has no path from s_i to t_i for any $i \in [p]$. We say that S is an independent multicut of T in G, if S is an independent set in G and S is a multicut of T in G. In this section we prove that the problem of finding an independent multicut (formally defined below) in a bounded degeneracy graph is FPT when parameterized by the solution size.

Stable Multicut	**Parameter:** k
Input: An undirected graph G, a set of terminal pairs T and $k \in \mathbb{N}$.	
Question: Is there an independent multicut of T in G of size at most k?	

Using Lemmas 3.2 and 3.3, and the known algorithm for Multicut [Lokshtanov et al. 2016a; Marx 2006] (the problem where we do not demand that the multicut to be an independent set), we prove the main theorem of this section. Before that, we state an algorithmic result for Multicut that is crucially used by our algorithm.

Lemma 4.4 ([Lokshtanov et al. 2016a; Marx 2006]). Multicut *can be solved in* $2^{\mathcal{O}(k^3)} \cdot mn \log n$ *time.*

We now define the annotated version of the Multicut problem, the way we defined it for the previously considered problems.

Annotated Multicut	**Parameter:** k
Input: An undirected graph G, a set of terminal pairs T, $Y \subseteq V(G)$ and $k \in \mathbb{N}$.	
Question: Is there a multicut S of T in G of size at most k such that $S \subseteq Y$?	

The following lemma give an algorithm for solving Annotated Multicut using the algorithm of Lemma 4.4.

Lemma 4.5. Annotated Multicut *can be solved in time* $2^{\mathcal{O}(k^3)} \cdot mn \log n$.

Proof sketch. We first give a polynomial time reduction from Annotated Multicut to Multicut which is described below.

Let (G, T, Y, k) be an instance of Annotated Multicut. Construct a graph G' from G by replace each $v \in V(G) \setminus Y$, with $k+1$ vertices $v_1, \ldots v_{k+1}$ with same neighborhood as v. That is, the neighborhood of $v_1, \ldots v_{k+1}$ are same in the resulting graph G'. We call the set of vertices that are added for v in G' as the block for v. We now construct the set of terminal pairs T' from the set of terminals T as follows. If $\{s,t\} \in T$ and $\{s,t\} \subseteq Y$, we add $\{s,t\}$ to T'. Suppose $\{s,t\} \in T$ and $\{s,t\} \cap Y = \{t\}$. Let $s_1, \ldots, s_{k+1}$ be the bock for s in G'. We add $\{s_1,t\}, \ldots \{s_{k+1},t\}$ to T'. Suppose $\{s,t\} \in T$ and $\{s,t\} \subseteq V(G) \setminus Y$. Let $s_1, \ldots, s_{k+1}$ and $t_1, \ldots, t_{k+1}$ be the blocks for s and t, respectively. We add $\{\{s_i, t_j\} \mid i,j \in [k+1]\}$ to T'.

We will now show that (G, T, Y, k) is a Yes instance of Annotated Multicut if and only if (G', T', k) is a Yes instance of Multicut. For the forward direction, let C be a multicut of size at most k in G such that $C \subseteq Y$. We claim that C is a multicut of T' in G'. Suppose not. Then, there is a path from s' to t' in $G' - C$, where $\{s', t'\} \in T'$. Let s and t be the vertices in $V(G)$ such that s' and t' are the vertices corresponding to them, respectively, that is, if $s' \in Y$, then $s = s'$, otherwise let s be the vertex such that s' is in the block of vertices constructed for the replacement of s in G'. By replacing each vertex in the $s' - t'$ path in G' by the corresponding vertex in G, we get a walk from s to t in $G - C$, which contradicts the fact that C is a multicut of T in G. For the backward direction, suppose C' is a minimal multicut of T' in G' of size at most k. Since, for any $v \in V(G) \setminus Y$, the neighborhood of $v_1, \ldots v_{k+1}$ in G' is the same as that of v in G and $|C'| \leqslant k$, $C' \cap \{v_1, \ldots, v_{k+1}\} = \emptyset$. Thus, $C' \subseteq Y$. Since G' is a supergraph of G and $T \subseteq T'$, C' is a multicut of T in G.

Thus, to find a k sized multicut of T in G which is fully contained in Y, it is enough to find a multicut of T' in G'. The total number of vertices in G' is at most $k|V(G)|$ and the total number of edges in G' is at most $(k+1)^2|E(G)|$. Thus, the running time of the algorithm follows from Lemma 4.4. This completes the proof sketch of the lemma. □

THEOREM 8. Stable Multicut *can be solved in time*

$$\min\{2^{\mathcal{O}(k^3)} \cdot \binom{k(1+d)}{k} \cdot 2^{o(k(1+d))} \cdot mn\log^2 n, 2^{\mathcal{O}(k^3)} \cdot \binom{k^2(1+d)^2}{k} \cdot d^{\mathcal{O}(1)} \cdot mn\log^2 n\}.$$

PROOF. Let (G, T, k) be an instance of STABLE MULTICUT. The algorithm first computes an independent set covering family $\mathcal{F}(G, k)$ using the algorithm of Lemma 3.2 or Lemma 3.3. By the definition of independent set covering family, any independent set S of G of cardinality at most k is contained in an independent set $I \in \mathcal{F}(G, k)$. In particular, if (G, T, k) is a Yes instance, then there is a solution which is fully contained in some $I \in \mathcal{F}(G, k)$ (moreover, the set I is independent in G). Therefore, to test whether (G, T, k) is a Yes instance or not, it is enough to test whether there is multicut (not necessarily independent) of size at most k contained in some $I \in \mathcal{F}(G, k)$. Hence, for each $I \in \mathcal{F}(G, k)$, the algorithm runs the algorithm of Lemma 4.5, and check whether there is a multicut of size at most k in I or not. If even one application of the algorithm of Lemma 4.5 gives a positive answer, the algorithm outputs Yes, otherwise it outputs No. The running time of this algorithm follows from Lemmas 3.2, 3.3 and 4.5. □

5 APPLICATIONS II: GENERAL GRAPHS

In this section, we solve STABLE s-t SEPARATOR and STABLE ODD CYCLE TRANSVERSAL on general graphs. The core of our algorithms is the Treewidth Reduction Theorem of [Marx et al. 2013] and our algorithms for SSTS and SOCT on bounded degeneracy graphs from Sections 4.1 and 4.2, respectively. We begin by stating the Treewidth Reduction Theorem.

THEOREM 9 (TREEWIDTH REDUCTION THEOREM, THEOREM 2.15 [MARX ET AL. 2013]). *Let G be a graph, $T \subseteq V(G)$ and $k \in \mathbb{N}$. Let C be the set of all vertices of G participating in a minimal s-t-separator of cardinality at most k for some $s, t \in T$. For every k and $|T|$, there is an algorithm that computes a graph G^* having the following properties, in time $2^{(k+|T|)^{\mathcal{O}(1)}} \cdot (n + m)$.*

(1) $C \cup T \subseteq V(G^*)$,
(2) for every $s, t \in T$, a set $K \subseteq V(G^*)$ with $|K| \leqslant k$ is a minimal s-t-separator of G^* if and only if $K \subseteq C \cup T$ and K is a minimal s-t-separator of G,
(3) the treewidth of G^* is at most $2^{(k+|T|)^{\mathcal{O}(1)}}$, and
(4) $G^*[C \cup T]$ *is isomorphic to* $G[C \cup T]$.

We remark here that Theorem 2.1 in [Marx et al. 2013] does not state explicit dependency on k and $|T|$ in both, the running time of the algorithm and the treewidth of G^* obtained.

Stable $s - t$ Separator. Let (G, k) be an instance of SSTS. To solve SSTS on general graphs, we first apply the Treewidth Reduction Theorem (Theorem 9) on $G, T = \{s, t\}$ and k to obtain a graph G^* with treewidth is upper bounded by . We then show that for SSTS, it is enough to work with this new graph G^* whose treewidth is bounded $2^{k^{\mathcal{O}(1)}}$. By conditions 2 and 4, to find a minimal independent s-t-separator separator in G, it is enough to a minimal independent s-t-separator in G^*. By Proposition 3.3, we know that the degeneracy of G^* is at most $2^{k^{\mathcal{O}(1)}}$, and hence we apply Theorem 2 or Theorem 3 to get a solution of SSTS on (G, k). That is, we get the following theorem.

THEOREM 10. *There is a randomized algorithm that solves* SSTS *in time* $2^{k^{\mathcal{O}(1)}}(n + m)$ *with success probability at least $1 - \frac{1}{e}$. There is a deterministic algorithm that solves* SSTS *in time* $2^{k^{\mathcal{O}(1)}}(n + m)\log n$.

Stable Odd Cycle Transversal. By using the Theorem 10 and Proposition 4.1 we get a $2^{k^{\mathcal{O}(1)}}(n+m)$ time (FPT linear time) algorithm for SOCT. Towards that, in the Theorem 4.2 of Marx et al. [Marx and Razgon 2014] we replace the algorithm of Kawarabayashi and Reed [ichi Kawarabayashi and Reed 2010] with Proposition 4.1 and the algorithm for SSTS with Theorem 10. For completeness we include the proof here.

Proposition 5.1 (Lemma 4.1, [Marx et al. 2013]). *Let G be a bipartite graph and let (B', W') be a proper 2-coloring of the vertices. Let B and W be two subsets of $V(G)$. Then, for any $S \subseteq V(G)$, the graph $G - S$ has a 2-coloring where $B \setminus S$ is black and $W \setminus S$ is white if and only if S separates $X := (B \cap B') \cup (W \cap W')$ and $Y := (B \cap W') \cup (W \cap B')$.*

Theorem 11. *There is a randomized algorithm that solves* SOCT *in time $2^{k^{\mathcal{O}(1)}}(n+m)$ with success probability at least $1 - \frac{1}{e}$. There is a deterministic algorithm that solves* OCT *in time $2^{k^{\mathcal{O}(1)}}(n+m)\log n$.*

Proof. Using the algorithm of Proposition 4.1, find a set $S_0 \subseteq V(G)$ of size at most k such that $G \setminus S_0$ is a bipartite graph. Observe that if such a set does not exist then (G, k) is No instance of SOCT. Thus, henceforth, we can assume that such a set S_0 exists. We next branch into $3^{|S_0|}$ cases, where each branch has the following interpretation. If we fix a hypothetical solution S and a proper 2-coloring of $G - S$, then each vertex of S_0 is either removed (that is, belongs to S, a fixed hypothetical solution), colored with the first color, say black, or colored with the second color, say white. For a particular branch, let R be the the vertices of S_0 to be removed (in order to get the hypothetical solution S) and let B_0 (respectively W_0) be the vertices of S_0 getting color black (respectively white) in a proper 2-coloring of $G - S$. A set S is said to be compatible with the partition (R, B_0, W_0), if $S \cap S_0 = R$ and $G \setminus S$ has a proper 2-coloring, with colors black and white, where the vertices in B_0 are colored black and the vertices in W_0 are colored white. Observe that (G, k) is a Yes instance of SOCT, if and only if for at least one branch corresponding to a partition (R, B_0, W_0) of S_0, there is a set S compatible with (R, B_0, W_0) of size at most k and S is an independent set. Note that we need to check only those branches corresponding to the partition (R, B_0, W_0) where $G[B_0]$ and $G[W_0]$ are edgeless graphs.

The next step is to transform the problem of finding a set compatible with (R, B_0, W_0) into a separation problem. Let (B', W') be a 2-coloring of $G - S_0$. Let $B = N(W_0) \setminus S_0$ and $W = N(B_0) \setminus S_0$. Let X and Y be the sets as defined in Proposition 5.1. That is, $X = (B \cap B') \cup (W \cap W')$ and $Y = (B \cap W') \cup (W \cap B')$. Construct a graph G' that is obtained from G by deleting the set $B_0 \cup W_0$, adding a new vertex s adjacent with $X \cup R$ and adding a new vertex t adjacent with $Y \cup R$. Notice that every s-t-separator in G' contains R. By Proposition 5.1, a set S is compatible with (R, B_0, W_0) if and only if S is an $s - t$ separator in G. Thus, we need to decide whether there is an s-t-separator S of size at most k such that $G'[S] = G[S]$ is an edgeless graph and this step can be done by Theorem 10.

Towards the run time analysis, we run the algorithm of Proposition 4.1 once, which takes time $2^{\mathcal{O}(k)}(m+n)$. Then we apply Theorem 10 at most 3^k times. Thus, we get the required running time. ◻

6 TOOL II: MULTICUT COVERING GRAPH SPARSIFICATION

This section starts by showing how to efficiently find some vertices that are irrelevant to "small" digraph pair cuts (defined in Section 6.1), assuming that the input graph has a sufficiently large number of vertices that are in-neighbors of the root. Afterwards, having a method to identify such irrelevant vertices at hand, we develop (in Section 6.2) an efficient algorithm that given a graph G, a set of terminal pairs T and a positive integer k, outputs an induced subgraph $G^\star$ of G and a subset $T^\star \subseteq T$ such that the following conditions are satisfied. First, any set $S \subseteq V(G)$ of size at most k is a minimal multicut of T in G if and only if $S \subseteq V(G^\star)$ and it is a minimal multicut of $T^\star$ in $G^\star$. Second, $G^\star$ does not contain any "large" $(k+2)$-connected set. Using this algorithm, we later give an FPT algorithm for Stable Multicut on general graphs.

6.1 Vertices Irrelevant to Digraph Pair Cuts

The notion of a digraph pair cut was defined by Kratsch and Wahlström in [Kratsch and Wahlström 2012]. This notion was used to derive randomized polynomial kernels for many problems, including Almost 2-SAT and Multiway Cut with Deletable Terminals. Towards defining which vertices are irrelevant to "small" digraph pair cuts, we first formally define what is a digraph pair cut.

Definition 6.1. *Let D be a digraph, T be a set of pairs of vertices (called* terminal pairs*), and $r \in V(D)$. We say that $S \subseteq V(D) \setminus \{r\}$ is an* r-T-digraph pair cut *if for every terminal pair $\{s,t\} \in T$, S is an s-r-separator or a t-r-separator.*[3]

The problem Digraph Pair Cut takes as input a digraph D, a set of terminal pairs T, $r \in V(D)$ and $k \in \mathbb{N}$, and the task is to output Yes if and only if there is an r-T-digraph pair cut in G of size at most k. We say that a vertex $v \in V(D)$ is *irrelevant* to the instance (D,T,r,k) if there is no minimal r-T-digraph pair cut of size at most k in D that contains v. If a vertex is not irrelevant to (D,T,r,k), then we say that it is *relevant* to (D,T,r,k). In the following lemma, which is the main result of this subsection, we show that for an instance (D,T,r,k) of Digraph Pair Cut, the number of in-neighbors of r that belong to at least one minimal r-T-digraph pair cut of size at most k is upper bounded by $64^{k+1}(k+1)^2$. In other words, we bound the number of in-neighbors of r that are relevant.

Lemma 6.1. *Let (D,T,r,k) be an instance of* Digraph Pair Cut*. The number of vertices in $N^-_D(r)$ that are relevant to (D,T,r,k) is at most $64^{k+1}(k+1)^2$. Moreover, there is a deterministic algorithm that given (D,T,r,k), runs in time $\mathcal{O}(|T| \cdot n(n^{\frac{2}{3}}+m))$, and outputs a set $R \subseteq N^-_D(r)$ of size at most $64^{k+1}(k+1)^2$ which contains all relevant vertices to (D,T,r,k) in $N^-_D(r)$.* [4]

Towards the proof of Lemma 6.1, we first define which terminal pairs are irrelevant.

Definition 6.2. *Let (D,T,r,k) be an instance of* Digraph Pair Cut*. A terminal pair $\{s,t\} \in T$ is* irrelevant *to (D,T,r,k) if any minimal r-$(T \setminus \{\{s,t\}\})$-digraph pair cut in D of size at most k is also a minimal r-T-digraph pair cut in D.*

The following observation directly follows from the definition of irrelevant terminal pairs.

Observation 6.1. *Let D be a digraph, T be a set of terminal pairs, $r \in V(D)$ and $k \in \mathbb{N}$. If $\{s,t\} \in T$ is a terminal pair irrelevant to (D,T,r,k), then any vertex relevant to (D,T,r,k) is also a vertex relevant to $(D, T \setminus \{\{s,t\}\}, r, k)$.*

We now define *important separators*, which have played an important role in the context of existing literature concerning cut related problems.

[3]The definition of digraph pair cut used here is same as that of Kratsch and Wahlström [Kratsch and Wahlström 2012] where we reverse the directions of the arcs of the graph.
[4]In other words, the vertices in $N^-_D(r) \setminus R$ are irrelevant are irrelevant to (D,T,r,k).

DEFINITION 6.3 (IMPORTANT SEPARATORS, [MARX 2006]). *Let* D *be a digraph. For subsets* $X, Y, S \subseteq V(D)$*, the set of vertices reachable from* $X \setminus S$ *in* $D - S$ *is denoted by* $R_D(X, S)$*. An* X-Y*-separator* S dominates *an* X-Y*-separator* S' *if* $|S| \leqslant |S'|$ *and* $R_D(X, S') \subset R_D(X, S)$*. A subset* S *is an* important X-Y-separator *if it is minimal, and there is no* X-Y*-separator* S' *that dominates* S*. For two vertices* $s, t \in V(D)$*, the term* important s-t-separator *refers to an important* $N^+_D(s)$-$N^-_D(t)$*-separator in* $D - \{s, t\}$*. For* $r \in V(D)$ *and* $Y \subseteq V(D)$*, the term* important r-Y-separator *refers to an important* $N^+_D(r)$-Y*-separator in* $D - r$*.*

LEMMA 6.2 ([CHEN ET AL. 2009; MARX 2006]). *Let* D *be a digraph,* $X, Y \subseteq V(D)$*, and* $k \in \mathbb{N}$*. The number of important* X-Y*-separators of size at most* k *is upper bounded by* 4^k*, and these separators can be enumerated in time* $\mathcal{O}(4^k \cdot k \cdot (n + m))$*.*

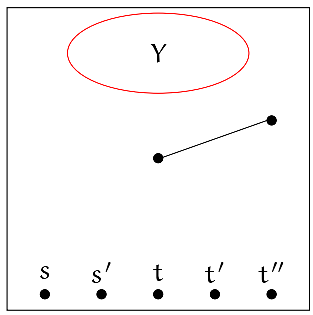
The rest of this subsection is dedicated to the proof of Lemma 6.1. That is, we design an algorithm, called $\mathcal{A}$, that finds a set R with the properties specified by Lemma 6.1. If $|N^-_D(r)| \leqslant 64^{k+1}(k+1)^2$, then $N^-_D(r)$ is the required set R. Thus, from now on, we assume that $|N^-_D(r)| > 64^{k+1}(k+1)^2$. Algorithm $\mathcal{A}$ is an iterative algorithm. In each iteration, $\mathcal{A}$ either terminates by outputting the required set R, or finds an irrelevant terminal pair for the input instance, removes it from the set of terminal pairs, and then repeats the process.

As a preprocessing step preceding the first call to $\mathcal{A}$, we modify the graph D and the set of terminal pairs T as described below. The new graph D' and set of terminal pairs T' would allow us to accomplish our task while simplifying some arguments in the proof. We set D' to be the digraph obtained from D by adding two new vertices, s' and t', and two new edges, $s's$ and $t't$, for each terminal pair $\{s, t\} \in T$. The modification is such that if a vertex $u \in V(D)$ belonged to ℓ terminal pairs in T, then D' would have ℓ distinct vertices corresponding to u. Now, the new set of terminal pairs is defined as $T' = \{\{s', t'\} \mid \{s, t\} \in T\}$. It is easy to see that any minimal r-T-digraph pair cut in D is also a minimal r-T'-digraph pair cut in D'. Thus, to find a superset of relevant vertices for (D, T, r, k) in the set $N^-_D(r)$, it is enough to find a superset of relevant vertices for (D', T', r, k) in the set $N^-_{D'}(r)$. Therefore, from now on we can assume that our input instance is (D', T', r, k), where the set T' is pairwise disjoint (see Figures 2b and 2c for an illustration). Henceforth, whenever we say that a vertex is relevant (or irrelevant), we mean that it is relevant (or irrelevant) for the instance (D', T', r, k). The description of $\mathcal{A}$ is given in Algorithm 2

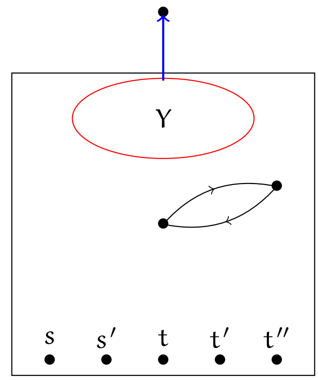
LEMMA 6.3. *Algorithm 2 outputs a set* R *of size at most* $64^{k+1}(k+1)^2$*, which contains all relevant vertices in* $N^-_D(r)$*.*

PROOF. Notice that Algorithm 2 returns a set R either in Line 2 or in Line 8 thus, by Lemma 6.2, the size of the returned set is at most $|Z| \cdot 4^k k + |Z| \leqslant 64^{k+1}(k+1)^2$. We now prove the correctness of the algorithm using induction on $|T'|$. When $|T'| = 0$, then no vertex in $N^-_D(r)$ is relevant and the algorithm returns the correct output. Now consider the induction step where $|T'| > 0$. We have two cases based on the size of the separator Z computed in Line 4.

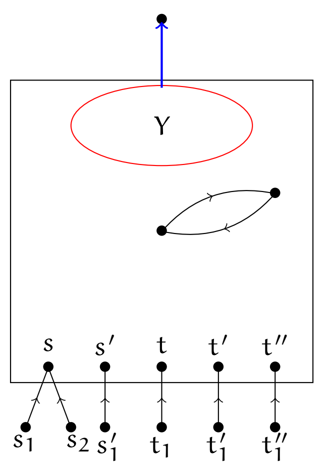
Case 1: $|Z| \leqslant 16^k \cdot 64(k+1)$. In this case, Lines 6-8 will be executed and Algorithm 2 will output a set R. We prove that R contains all relevant vertices in $N^-_{D'}(r)$. Towards this, we show that if S is a minimal r-T'-digraph pair cut of size at most k and $v \in N^-_{D'}(r) \cap S$, then v belongs to R. Let $S' = S \setminus \{v\}$. Since S is a minimal r-T'-digraph pair cut, S' is not a r-T'-digraph pair cut. Since S is a r-T'-digraph pair cut and S' is not a r-T'-digraph pair cut, there is a vertex $t \in \widehat{T}$ such that (i) v is reachable from t in $D' - S'$, and (ii) r is not reachable from t in $D' - S$. If $v \in Z$, then v is marked and belongs to R. Therefore, if $v \in Z$, we are done. Thus, from now on, assume that $v \notin Z$.



(a) Let Y be a $(k+1)$-connected set in G.



(b) Constructing D from G. Here, $Y = N^{-}(r)$.



(c) Constructing D' from D'.

Fig. 2. The graphs G, D and D' are displayed in left-to-right order, $T = \{\{s,t\},\{s,t''\},\{s',t'\}\}$ and $T' = \{\{s_1,t_1\},\{s_2,t_1''\},\{s_1',t_1'\}\}$.

Claim 6.1. *There is a vertex $z \in Z$ that belongs to $R_{D'}(t,S)$.*

Proof. From (i), we have that $v \in R_{D'}(t,S')$. Since Z is a minimum $\widehat{T}$-r-separator, $t \in \widehat{T}$, and $v \in R_{D'}(t,S')$, we have that all paths from t to v passes through some vertex in Z. Also, since $v \in N^{-}_{D'}(r)$ and $v \in R_{D'}(t,S')$ and $v \notin Z$, there is a vertex $z \in Z$ that belongs to $R_{D'}(t,S)$. ◻

Let $R_t = R_{D'}(t,S)$ and $C = N^{+}_{D'}(R_t)$. Observe that $C \subseteq S$, $v \in C$ and v is reachable from z in $D' - (C \setminus \{v\})$. We claim that C is a z-r-separator. If C is not a z-r-separator, then there is a path from z to r in $D' - S$. Also, since $z \in R_{D'}(t,S)$, there is a path from t to z in $D' - S$. This implies that there is a path from t to r in $D' - S$ which is a contradiction to the statement (ii). Since v is reachable from z in $D' - (C \setminus \{v\})$, there is a minimal z-r-separator that contains v and is fully contained in C. Let $C' \subseteq C$ be a minimal z-r-separator that contains v. Since $v \in N^{-}_{D'}(r)$ and C' is a minimal z-r-separator, either C' is an important

Algorithm 2: Input is (G', T', r, k), where T' is pairwise disjoint

1 **if** $|T'| = 0$ **then**
2   **return** $\emptyset$
3 $\widehat{T} := \{s', t' \mid \{s', t'\} \in T'\}$.
4 Compute a minimum $\widehat{T}$-r-separator Z.
5 **if** $|Z| \leqslant 16^k \cdot 64(k+1)$ **then**
6   For each $z \in Z$, compute all important z-r-separators of size at most k.
7   Mark all the vertices in $N^-_{D'}(r)$, which are either part of the computed important separators or part of Z.
8   **return** *the set of marked vertices (call it* R*)*
9 **else**
10   Compute a maximum set $\mathcal{P}$ of vertex disjoint paths from $\widehat{T}$ to r (any pair of paths intersects only at r).
11   Let $X = V(\mathcal{P}) \cap \widehat{T}$. Let A be a maximum sized subset of X such that for any $\{s', t'\} \in T'$, $|A \cap \{s', t'\}| \leqslant 1$.
12   Let $B = \{w \mid$ there exists $w' \in A$ such that $\{w, w'\} \in T'\}$. That is, B is the set of vertices that are paired with vertices of A in the set of pairs T'.
13   Compute all important r-B-separators of size at most $2k+2$ in $\overleftarrow{D}'$.
14   Mark all vertices from B which are part of the computed important separators.
15   Let $q \in B$ be an unmarked vertex and let $\{q, q'\} \in T'$.
16     $T' := T' \setminus \{\{q, q'\}\}$ and repeat from Step 2.

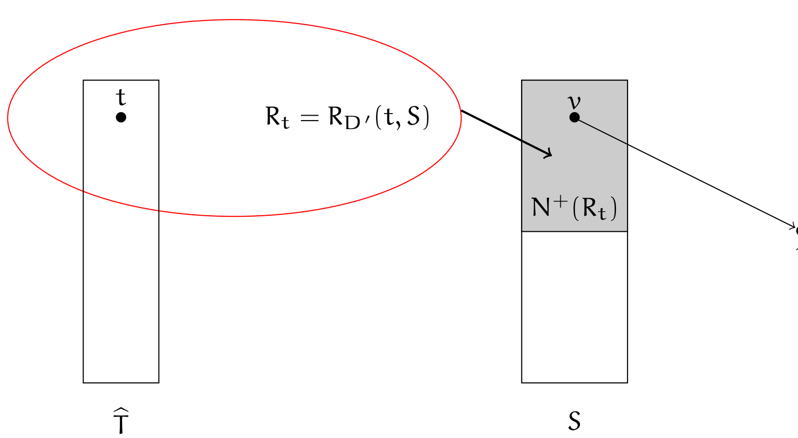


Fig. 3. Here, the ellipse contains the set of vertices reachable from t in $D' - S$, denoted by R_t. The rectangle colored grey represents $N^+(R_t)$ which includes v

z-r-separator or there is an important z-r-separator of size at most k containing v which dominates C'. In either case, v is marked in Line 8 and hence, it will be in the set R (see Figure 3 for an illustration).

Case 2: $|Z| > 16^k \cdot 64(k+1)$. In this case, we prove that there, indeed, exists an unmarked vertex $q \in B$ and the pair $\{q, q'\}$ is an irrelevant terminal pair. Notice that in Line 13, we have computed all important r-B-separators of size at most $2k+2$ for some B. By Lemma 6.2, the total number of vertices in all these separators together is at most $16^k \cdot 32(k+1)$. So we should have marked at most $16^k \cdot 32(k+1)$ vertices in B. We first claim that $|B| > 16^k \cdot 32(k+1)$, which ensures the existence of an unmarked vertex in B. By the definition of A, the size of A is at least $|Z|/2 > 16^k \cdot 32(k+1)$, because there are $|Z|$ vertex disjoint paths from $\widehat{T}$ to r, only intersecting at r. By the definition of B, $|B| = |A| > 16^k \cdot 32(k+1)$. Since we proved that we have only marked at most $16^k \cdot 32(k+1)$ vertices in B, this implies that there is an unmarked vertex q in B. Let q' be the unique vertex such that $\{q, q'\} \in T'$ (such a unique vertex exists because T' is pairwise disjoint).

Now we show that $\{q, q'\}$ is an irrelevant terminal pair. Let S be a minimal r-$(T' \setminus \{\{q, q'\}\})$-digraph pair cut of size at most k. We need to show that S is also a r-T'-digraph pair cut. We know that there are $|Z|$ vertex disjoint paths $\mathcal{P}$ from $\widehat{T}$ to r, where the paths intersect only at r. Since Z is a minimum $\widehat{T}$-r-separator, $|\widehat{T}| \geqslant |Z|$. Recall the definition of A and B from the description of the algorithm. Let A_r be the set of vertices in $A \setminus \{q'\}$ such that r is reachable from each vertex in A_r in $D' - S$, that is, $A_r = \{u \in A \setminus \{q'\} \mid r \in R_{D'}(u, S)\}$. Let B_r is the set of vertices in B such that r is reachable from any vertex in B in $D' - S$, that is, $B_r = \{u' \in B \mid r \in R_{D'}(u', S)\}$. Since there are $|A|$ vertex disjoint paths from A to r (which intersect only at r) and $|S| \leqslant k$ $|A_r| \geqslant |A| - (k+1)$. Since S is an r-$(T' \setminus \{\{q, q'\}\})$-digraph pair cut, the vertices in B which are paired with a vertex in A_r are not reachable from r in $\overleftarrow{D'} - S$. This implies that $|B_r| \leqslant k+1$. Let $Q = S \cup B_r \cup \{q\}$. Notice that $q \in Q$ and Q is a r-B-separator in $\overleftarrow{D'}$ of size at most $2k+2$. If q is not reachable from r in $\overleftarrow{D'} - S$, then S is, indeed, a r-T'-digraph pair cut, because S is a r-$(T' \setminus \{\{q, q'\}\})$-digraph pair cut. In what follows we show that it is always the case, that is, q is not reachable from r in $\overleftarrow{D'} - S$. Suppose not. Since q is reachable from r in $\overleftarrow{D'} - S$ and all the vertices in $Q \setminus S$ have no out-neighbors in $\overleftarrow{D'}$ (by construction of D'), any path from r to q in $\overleftarrow{D'} - S$ will not contain any vertex from $Q \setminus \{q\}$. This implies that there is a minimal r-B-separator $Q' \subseteq Q$ containing q. Hence, either Q' is an important r-B-separator of size at most $2k+2$ or all the important r-B-separators which dominate Q' will contain q. This is implies that q is marked, which is a contradiction.

Thus, we have shown that in this case there is an irrelevant terminal pair $\{q, q'\} \in T'$, and by ObservationObservation 6.1 and induction hypothesis, Algorithm 2 will output the required set. ◻

Lemma 6.4. *Algorithm 2 runs in time* $\mathcal{O}(|T'| \cdot n(n^{\frac{2}{3}} + m))$.

Proof. The number of times each step of the algorithm will get executed is at most $|T'|$. By Proposition 2.2, Line 4 takes time $\mathcal{O}(mn)$. By Lemma 6.2, the time required to enumerate important separators in Lines 6 and 13 is bounded by $\mathcal{O}(4^{2k} \cdot k \cdot (n+m))$. The time required compute $\mathcal{P}$ in Line 13 is $\mathcal{O}(mn)$ by Proposition 2.2. Thus, the total running time of Algorithm 2 is $\mathcal{O}(|T'|(mn + 4^{2k} \cdot k \cdot (n+m)))$. Recall that, we could safely assume that $|V(D')| = n > 64^{k+1}(k+1)^2$. Since, $n > 64^{k+1}(k+1)^2$, $4^{2k} \cdot k < n^{\frac{2}{3}}$. Hence, the claimed running time of the algorithm follows. ◻

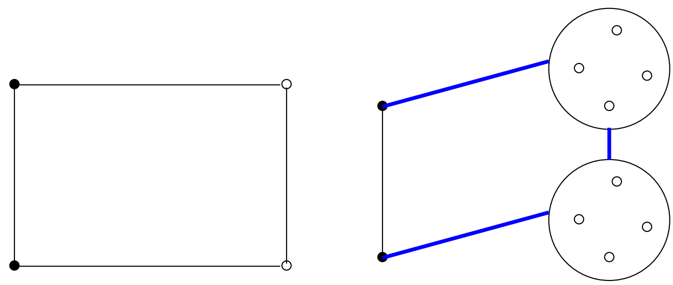


Fig. 4. The graph at the right hand side is obtained by the reduction on the graph at the left hand side, where $k = 3$ and Y is the set of black colored vertices. Thick lines represents all possible edges between two sets of vertices.

6.2 Covering Small Multicuts in a Subgraph without Highly Connected Set

In this section, we prove that given a graph G, a set of terminal pairs $T = \{\{s_1, t_1\}, \ldots, \{s_\ell, t_\ell\}\}$ and an integer k, there is a polynomial time algorithm which finds a pair $(G^\star, T^\star)$, where $G^\star$ is an induced subgraph of G such that it has no $(k+2)$-connected sets of size $2^{\mathcal{O}(k)}$ and $T^\star \subseteq T$ such that for any $S \subseteq V(G)$ of size at most k, S is a minimal multicut of T in G if and only S is a subset of $V(G^\star)$ and S is a minimal multicut of $T^\star$ in $G^\star$. This statement is formalized in Lemma 6.5. Before stating Lemma 6.5, we give definitions of a k-connected set in a graph G and a k-connected graph.

Definition 6.4 (k-connected set and graphs). *For any $k \in \mathbb{N}$ and a graph G, a subset Y of the vertices of G is called a* k-connected set *in G, if for any $u, v \in Y$ there are at least k internally vertex disjoint paths from u to v in G. The graph G is called a* k-connected graph *if $V(G)$ is a k-connected set in G. Equivalently, the graph G is k-connected, if the size of a mincut in G is at least k.*

Lemma 6.5 (Degeneracy Reduction Lemma). *Let G be a graph, T be a set of terminal pairs and $k \in \mathbb{N}$. Let C be the set of all minimal multicuts of T of size at most k in G. There is a deterministic algorithm which runs in time $\mathcal{O}(|T| \cdot n^2(n^{\frac{2}{3}} + m) + kn^3(n+m))$ and outputs an induced subgraph $G^\star$ of G and a subset $T^\star \subseteq T$ such that*

(1) for any $S \subseteq V(G)$ with $|S| \leqslant k$, S is a minimal multicut of T in G if and only if $S \subseteq V(G^\star)$ and S is a minimal multicut of $T^\star$ in $G^\star$, and
(2) there is no $(k+2)$-connected set of size at least $64^{k+2} \cdot 4(k+2)^2$ in $G^\star$.

The proof of Lemma 6.5 requires some auxiliary lemmas which we discuss below. Recall the definition of the problem Multicut from Section 4.4. Let (G, T, k) be an instance of Multicut. We say that a vertex $v \in V(G)$ is *irrelevant* to (G, T, k) if no minimal multicut of G of size at most k in G contains v. Lemma 6.6 states that if a graph has a sufficiently large $(k+2)$-connected set, then many of its vertices are irrelevant to the given Multicut instance. Such a statement is deduced by establishing a relation between the multicuts of the given instance and the digraph pair cuts of practically the same instance. This relationship then relates the irrelevant vertices to the instance of Multicut with the irrelevant vertices to the instance for Digraph Pair Cut.

Lemma 6.6. *Let G be a graph, T be a set of terminal pairs, $k \in \mathbb{N}$ and Y be a $(k+1)$-connected set in G. Let D be a digraph obtained by adding a new vertex r, whose in-neighbors are the vertices of Y, and replacing each edge of G by two arc, with the same endpoints, in* *opposite orientations. Any irrelevant vertex of* Y *to the instance* (D, T, r, k) *of* Digraph Pair Cut *is also an irrelevant vertex to the instance* (G, T, k) *of* Multicut.

Proof. The construction of D from G is illustrated in Figures 2a and 2b. Suppose there exists $v \in Y$ which is relevant to the instance (G, T, k) of Multicut. Then, there exists a minimal multicut, say C, of T in G of size at most k such that $v \in C$. We first claim that C is an r-T-digraph pair cut in D. Suppose not. Then, there is a pair $\{s, t\} \in T$ such that there is a path from s to r and t to r in $D - C$. Since the in-neighbors of r are the vertices of Y, there exist $u_1, u_2 \in Y$, u_1 may be equal to u_2, such that there are two paths, one from s to u_1 and other from t to u_2, in $G - C$. If $u_1 = u_2$, then s and t are in the same connected component of $G - C$, which is a contradiction. Otherwise, since Y is a $(k+1)$-connected set in G and $u_1, u_2 \in Y$, there are $k+1$ internally vertex disjoint paths from u_1 to u_2. Since $|C| \leqslant k$, there exists a path between u_1 and u_2 in $G - C$, and hence a path between s and t in $G - C$, which is a contradiction.

We next show that there exists $C' \subseteq C$ such that $v \in C'$ and C' is a minimal r-T-digraph pair cut in D. This will prove that v is relevant to the instance (D, T, r, k) of Digraph Pair Cut thereby proving the claim. Since C is an r-T-digraph pair cut in D, there exists $C' \subseteq C$ such that C' is a minimal r-T-digraph pair cut in D. Suppose $v \notin C'$. Since C is a minimal multicut of T in G, there exists a terminal pair $(s, t) \in T$ such that there is a path from s to t in $G - \{C \setminus \{v\}\}$. In particular, there is a path from s to v and t to v in $G - \{C \setminus \{v\}\}$. Since $v \in Y$, by the construction of D, v is an in-neighbor of r. Hence, there is a path from s to r and t to r in $D - \{C \setminus \{v\}\}$. Thus, if $v \notin C'$, then C' is not an r-T-digraph pair cut in D, which is a contradiction. ◻

Using Lemmas 6.1 and 6.6, one can find irrelevant vertices to the given instance of Multicut, if the graph in the instance has a $(k+1)$-connected set Y of size strictly more than $64^{k+1}(k+1)^2$ and the set Y is explicitly given as input. So the next task is to design an algorithm that finds a $(k+1)$-connected set in a graph of a given size, if it exists. This algorithm comes from Lemma 6.7.

Lemma 6.7. *There is an algorithm which given a graph* G *and* $k, d \in \mathbb{N}$, $k \leqslant d$, *runs in time* $\mathcal{O}(k \cdot n^2(n+m))$, *and either concludes that there is no* k*-connected set of size at least* $4d$ *in* G *or outputs a* k*-connected set in* G *of size at least* $d+1$.

The proof of Lemma 6.7 requires an auxiliary lemma (Lemma 6.9) which we prove next. Lemma 6.9 is an algorithmic version of the following famous result of Mader [Mader 1972] which says that if a graph has large average degree (or degeneracy), then it contains a $(d+1)$-connected subgraph.

Lemma 6.8 ([Mader 1972]). *Let* $d \in \mathbb{N} \setminus \{0\}$. *Every graph* G *with average degree at least* $4d$ *has a* $(d+1)$*-connected subgraph.*

The proof of Lemma 6.8 given in [Diestel 2000; Sudakov 2016] can be modified to get a polynomial time algorithm. The following lemma, an algorithmic version of Lemma 6.8, is written in terms of the degeneracy of the graph.

Lemma 6.9. *There is an algorithm which, for any* $d \in \mathbb{N} \setminus \{0\}$, *given a graph* G *with degeneracy at least* $4d$, *runs in time* $\mathcal{O}(mn + n^2 \log n)$, *and outputs a* $(d+1)$*-connected subgraph of* G.

Proof. The algorithm first constructs a subgraph H of G which has minimum degree at least $4d$. To do so, first set $H := G$. If the minimum degree of H is at least $4d$, then we are done. Otherwise, let v be a vertex of H of degree at most $4d-1$. Set $H := H - v$ and repeat this process. Since the degeneracy of G is at least $4d$, the procedure will end up in a subgraph of G that has minimum degree at least $4d$. The naive implementation of the above procedure takes time $\mathcal{O}(mn)$.

Claim 6.2. *For any* $d \in \mathbb{N} - \{0\}$*, if the minimum degree of a graph* H *is at least* $4d$*, then* $|V(H)| \geqslant 2d+1$ *and* $|E(H)| \geqslant 2d(|V(H)| - d - \frac{1}{2})$.

Proof. Since minimum degree of H is at least $4d$, clearly $|V(H)| \geqslant 4d+1 \geqslant 2d+1$. Also, since ${}_{v\in V(H)}\, \mathsf{deg}_G(v) = 2|E(H)|$ and for all $v \in V(H)$ $\mathsf{deg}_G(v) \geqslant 4d$, $|E(H)| \geqslant 2d|V(H)| \geqslant 2d(|V(H)| - d - \frac{1}{2})$. ◻

From Claim 6.2, we conclude that $|V(H)| \geqslant 2d+1$ and $|E(H)| \geqslant 2d(|V(H)| - d - \frac{1}{2})$. Thus, from the following claim (Claim 6.3), one can infer that H has a $(d+1)$-connected subgraph. Using this claim, we will later give an algorithm that actually computes a $(d+1)$-connected subgraph of H, whose correctness will follow from the proof of Claim 6.3.

Claim 6.3. *Let* H *be any graph and* $d \in \mathbb{N} \setminus \{0\}$ *such that* $|V(H)| \geqslant 2d+1$ *and* $|E(H)| \geqslant 2d(|V(H)| - d - \frac{1}{2})$. *Then* H *has a* $(d+1)$*-connected subgraph.*

Proof. We prove the claim using induction on $|V(H)|$. The base case of the induction is when $|V(H)| = 2d+1$. From the premises of the claim, if $|V(H)| = 2d+1$, $|E(H)| \geqslant 2d(2d+1-d-\frac{1}{2}) = 2d(d+\frac{1}{2}) = \binom{2d+1}{2}$. Since a graph on $2d+1$ vertices can have at most $\binom{2d+1}{2}$ edges, H is a clique on $2d+1$ vertices, which is a $(d+1)$-connected graph. Now consider the induction step where $|V(H)| > 2d+1$. Suppose there is a vertex $v \in V(H)$ such that $\mathsf{deg}_H(v) \leqslant 2d$. Then $|V(H-v)| \geqslant 2d+1$ and $|E(H-v)| \geqslant |E(H)| - 2d \geqslant 2d(|V(H-v)| - d - \frac{1}{2})$. Thus, from the induction hypothesis, there is a $(d+1)$-connected subgraph in $H-v$. From now on, we can assume that the degree of each vertex in H is at least $2d+1$. Suppose H itself is a $(d+1)$-connected graph, then we are done. If not, then there exists a mincut, say Z, of H, of size at most $2d$. Let $U_1 \uplus U_2$ be a partition of $V(G) \setminus Z$ such that there is no edge between a vertex in U_1 and a vertex in U_2, and $U_1, U_2 \neq \emptyset$. Let $A = Z \cup U_1$ and $B = Z \cup U_2$. We claim that either $H[A]$ or $H[B]$ satisfy the premises of the claim. Notice that all the neighbors of any vertex $s \in U_1$ are in A and all the neighbors of any vertex $t \in U_2$ are in B. Also since, $\mathsf{deg}_H(s), \mathsf{deg}_H(t) \geqslant 2d+1$, we have that $|A| \geqslant 2d+1$ and $|B| \geqslant 2d+1$. Thus,the vertex set cardinality constraint stated in the premise of the claim is met for both $H[A]$ and $H[B]$. Suppose that, the edge set cardinality constraint stated in the premise of the claim is not met for both both $H[A]$ and $H[B]$. Then we have the following.

$$
\begin{aligned}
|E(H)| &\leqslant |E(H[A])| + |E(H[B])| \\
&< 2d(|A| - d - \frac{1}{2}) + 2d(|B| - d - \frac{1}{2}) \\
&= 2d(|A| + |B| - 2d - 1) \\
&\leqslant 2d(|V(H)| + d - 2d - 1) \\
&< 2d(|V(H)| - d - \frac{1}{2}).
\end{aligned}
$$

This is a contradiction to the fact that $|E(H)| \geqslant 2d(|V(H)| - d - \frac{1}{2})$. Therefore, either $H[A]$ or $H[B]$ satisfy the premises of the claim. Moreover, notice that $|A| < |V(H)|$ and $|B| < |V(H)|$, because $U_1, U_2 \neq \emptyset$. Thus, by the induction hypothesis the claim follows. ◻

The above proof can easily be turned in to an algorithm. This is explained below. Our algorithm for finding a $(d+1)$-connected subgraph of H works as follows. It first tests whether H itself is a $(d+1)$-connected graph - this can be done by computing a mincut of H (using the algorithm of Proposition 2.3) and then testing whether the size of a mincut of H is at least $d+1$. If H is a $(d+1)$-connected graph, then our algorithm outputs H. Otherwise, if there is a vertex of degree at most $2d$ in H, then it recursively finds a $(d+1)$-connected subgraph in $H-v$. If all the vertices in H have degree at least $2d+1$, then it finds a mincut Z in H (using the algorithm of Proposition 2.3). It then constructs vertex sets A and B as mentioned in the proof of Claim 6.3. It is proved in Claim 6.3 that either $H[A]$ or $H[B]$ satisfy the premises of Claim 6.3, and it can be tested in linear time whether a graph satisfies the premises of Claim 6.3. If $H[A]$ satisfies the premises of Claim 6.3, then our algorithm recursively finds a $(d+1)$-connected subgraph in $H[A]$. Otherwise, our algorithm recursively find a $(d+1)$-connected subgraph in $H[B]$.

Note that this algorithm makes at most n recursive calls and in each recursive call it runs the algorithm of Proposition 2.3 and does some linear time testing. Thus, given a graph H of minimum degree at least $2d$, this algorithm runs in time $\mathcal{O}(n(m+n\log n))$ and outputs a $(d+1)$-connected subgraph of H. The algorithm claimed in the lemma first constructs a subgraph H of G of minimum degree at least $2d$, as described earlier, in time $\mathcal{O}(mn)$ and takes additional $\mathcal{O}(mn+n^2\log n)$ time to output a $(d+1)$-connected subgraph of H. Thus, the total running time of this algorithm is $\mathcal{O}(mn+n^2\log n)$. ◻

We are now equipped to give the proof of Lemma 6.7.

Proof of Lemma 6.7. The algorithm first constructs an auxiliary graph G^* as follows. The vertex set of G^* is $V(G)$ and for any $u,v \in V(G^*)$,$uv \in E(G^*)$ if and only if the size of a minimum u-v-separator in G is at least k (that is, there are at least k internally vertex disjoint paths from u to v in G). It then checks whether the degeneracy of G^* is at least $4d-1$ or not. If the degeneracy of G^* is strictly less than $4d-1$, then the algorithm outputs that there is no k-connected set in G of size at least $4d$. Otherwise, the degeneracy of G^* is at least $4d-1 \geqslant 4(d-1)$. In this case, the algorithm applies the algorithm of Lemma 6.9 for $(G^*, d-1)$, which outputs a d-connected subgraph H of G^*. Since H is a d-connected subgraph, $|V(H)| \geqslant d+1$. Since, $k \leqslant d$, H is k-connected in G^*. The algorithm outputs $V(H)$ as the k-connected set in G.

To prove the correctness of the algorithm, we need to prove the following two statements.

(1) When our algorithm reports that there is no k-connected set in G of size at least $4d$, that is, when degeneracy of G^* is at most $4d-2$, then the graph G has no k-connected set of size at least $4d$.

(2) When our algorithm outputs a set, that is, when degeneracy of G^* is at least $4d-1$, then the set outputted is a k-connected set in G of size at least $d+1$. In other words, if degeneracy of G^* is at least $4d-1$, then the set $V(H^*)$ is k-connected in G and has size at least $d+1$.

For the proof of the first statement, observe that when G has a k-connected set, say Y, of size at least $4d$, then $G^*[Y]$ is a clique. Hence, the degeneracy of G^* is at least $4d-1$. For the proof of the second second, we have already argued that the size of $V(H)$ is at least $d+1$ and that H is a k-connected subgraph in G^*. We will now prove $V(H)$ is a k-connected set in G.

Claim 6.4. *$V(H)$ is a k-connected set in G.*

Proof. Observe that, it is enough to show that for any $u, v \in V(H)$ and any $C \subseteq V(G) \setminus \{u, v\}$ of size strictly less than k, there is a path from u to v in $G - C$. Since H is a k-connected subgraph of G^*, there is a path from u to v in $G^* - C$. Let $w_1 w_2 \ldots w_\ell$, where $w_1 = u$ and $v = w_\ell$, be a path from w_1 to w_ℓ in $G^* - C$. Since for any $i \in [\ell - 1]$, $w_i w_{i+1} \in E(G^*)$, there are at least k vertex disjoint paths from w_i to w_{i+1} in G. Also, since $|C| < k$, there is a path from w_i to w_{i+1} in $G - C$. This implies that there is a path from $w_1 = u$ to $w_\ell = v$ in $G - C$, proving that H is a k-connected set in G. □

This finishes the proof of correctness of our algorithm. We now analyze the total running time of the algorithm. The graph G^* can be constructed in time $\mathcal{O}(k \cdot n^2(n+m))$ using Proposition 2.2. Also, checking whether the graph has degeneracy at least $4d - 1$ can be done in time $\mathcal{O}(mn)$. Since G^* could potentially have $\mathcal{O}(n^2)$ edges, by Lemma 6.9, the subgraph H can be computed in time $\mathcal{O}(n^3)$. Thus the total running time of our algorithm is $\mathcal{O}(k \cdot n^2(n+m))$. □

Lemma 6.10. *There is an algorithm that given a graph* G*, a set of terminal pairs* T *and* $k \in \mathbb{N}$*, runs in time* $\mathcal{O}(|T| \cdot n(n^{\frac{2}{3}} + m) + kn^2(n+m))$ *and, either correctly concludes that* G *does not contain a* $(k+1)$*-connected set of size at least* $64^{k+1} \cdot 4(k+1)^2$ *or finds an irrelevant vertex for the instance* (G, T, k) *of* Multicut.

Proof. Let $d = 64^{k+1}(k+1)^2$. Our algorithm first runs the algorithm of Lemma 6.7 on the instance $(G, k+1, d)$. If this algorithm (of Lemma 6.7) concludes that there is no $(k+1)$-connected set in G of size at least $4d$, then our algorithm returns the same. Otherwise, the algorithm of Lemma 6.7 outputs a $(k+1)$-connected set Y in G of size at least $d+1$. Our algorithm then creates a digraph D as mentioned in Lemma 6.6. It then applies the algorithm of Lemma 6.1 and compute a set Z of irrelevant vertices for the instance (D, T, r, k) of Digraph Pair Cut in the set Y. From Lemma 6.6, Z is also a set of irrelevant vertices for the instance (G, T, k) of Multicut. Since $|Y| \geqslant d+1$ and the number of relevant vertices for (D, T, r, k) in the set Y is at most d (from Lemma 6.1), $Z \neq \emptyset$. Our algorithm then outputs an arbitrary vertex v from the set Z as an irrelevant vertex for (G, T, k).

By Lemmas 6.1 and 6.7, the total running time of our algorithm is $\mathcal{O}(|T| \cdot n(n^{\frac{2}{3}} + m) + kn^2(n+m))$. □

Lemma 6.11. *There is an algorithm which given as input a graph* G*, a set of terminal pairs* T *and* $k \in \mathbb{N}$*, runs in time* $\mathcal{O}(|T| \cdot n(n^{\frac{2}{3}} + m) + kn^2(n+m))$ *and, either concludes that there is no* $(k+2)$*-connected set of size at least* $64^{k+2} \cdot 4(k+2)^2$ *in* G*, or outputs a vertex* $v \in V(G)$ *such that for any* $S \subseteq V(G)$ *with* $|S| \leqslant k$*,* S *is a minimal multicut of* T *in* G *if and only if* $S \subseteq V(G) \setminus \{v\}$ *and* S *is a minimal multicut of* $T' = \{\{s,t\} \in T \mid v \notin \{s,t\}\}$ *in* $G - v$.

Proof. This algorithm runs the algorithm of Lemma 6.10 on the instance $(G, T, k+1)$. If the algorithm of Lemma 6.10 outputs that there is no $(k+2)$-connected set of size $64^{k+2} \cdot 4(k+2)^2$ in G, then our algorithm reports the same. Otherwise, let v be the vertex returned by the algorithm of Lemma 6.10, which is irrelevant for $(G, T, k+1)$ (from Lemma 6.10), then it also returns v. The running time of our algorithm follows from Lemma 6.10.

We now prove the correctness of this algorithm. For the forward direction, $S \subseteq V(G)$ such that $|S| \leqslant k$ and S is a minimal multicut of T in G. By the definition of an irrelevant vertex for the instance $(G, T, k+1)$, we conclude that $S \subseteq V(G) \setminus \{v\}$. Since $T' \subseteq T$ and G is a supergraph of $G - v$, S is a multicut of T' in $G - v$. Suppose, for the sake of contradiction, that S is not a minimal multicut of T' in $G - v$. Then, there exists $S' \subset S$ such that S' is a minimal multicut of T' in $G - v$. If S' is multicut of T in G, then we contradict the fact that S is a minimal multicut of T in G. Otherwise, there exists $S'' \subseteq S \cup \{v\}$ and $v \in S''$, such that S'' is a minimal multicut of T in G, which contradicts that v is an irrelevant vertex for $(G, T, k + 1)$. Hence we have proved that S is a minimal multicut of T' in $G - v$.

For the backward direction, let $S \subseteq V(G) \setminus \{v\}$ such that S is a minimal multicut of T' in $G - v$. If S is a multicut of T in G, then S has to a minimal multicut of T in G else it would contradict that S is a minimal multicut of T' in $G - v$. Otherwise, $S \cup \{v\}$ is a multicut of T in G, because all the terminal pairs in $T \setminus T'$ contains v. Let $S' \subseteq S \cup \{v\}$ be a minimal multicut of T in G. Note that $v \in S'$ and $|S'| \leqslant k + 1$. This contradicts the fact that v is an irrelevant vertex for $(G, T, k + 1)$.

◻

Lemma 6.5 can easily be proved by applying Lemma 6.11 at most n times.

Stable Multicut on General Graphs. With the power of our Independent Set Covering Lemmas (Lemmas 3.1, 3.2 and 3.3) and the Degeneracy Reduction Lemma (Lemma 6.5) in hand, we are now ready we prove that Stable Multicut is FPT. Towards that, we first prove the following lemma which establishes a relationship between the degeneracy of the graph and the k-connected sets in the graph.

Lemma 6.12. *Let $k, d \in \mathbb{N}$ such that $k \leqslant d + 1$. Let G be a graph which does not contain a k-connected set of size at least d. Then, the degeneracy of G is at most $4d - 1$.*

Proof. For the sake of contradiction, assume that the degeneracy of G is at least $4d$. Then, by Lemma 6.9, there is a $(d + 1)$-connected subgraph H of G. Since $k \leqslant d + 1$ and $|V(H)| \geqslant d + 2$, we have that $V(H)$ is k-connected set in G of size at least $d + 2$, which is a contradiction. ◻

Theorem 12. Stable Multicut *can be solved in time $2^{\mathcal{O}(k^3)} \cdot n^3(n + m)$.*

Proof. Let (G, k) be an instance of Stable Multicut. First, we apply Lemma 6.5 and get an equivalent instance (G^*, T^*), where G^* does not contain any $(k + 2)$-connected set of size $64^{k+2} \cdot 4(k + 2)^2$. Then, by Lemma 6.12, the degeneracy of G^* is at most $64^{k+2} \cdot 16(k + 2)^2 - 1$. Now, we apply Theorem 8 and get the solution. The running time of the algorithm follows from Lemma 6.5 and Theorem 8. ◻

7 CONCLUSION

In this paper we presented two new combinatorial tools for the design of parameterized algorithms. The first was a simple linear time randomized algorithm that given as input a d-degenerate graph G and integer k, outputs an independent set Y, such that for every independent set X in G of size at most k the probability that X is a subset of Y is at least $\left(\binom{(d+1)k}{k} \cdot k(d + 1)\right)^{-1}$. We also introduced the notion of a k-*independence covering family* of a graph G. The second tool was a new (deterministic) polynomial time graph sparsification procedure that given a graph G, a set $T = \{\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_\ell, t_\ell\}\}$ of terminal pairs, and an integer k returns an induced subgraph $G^\star$ of G that maintains *all* of the inclusion minimal multi-cuts of G of size at most k, and does not contain any $(k + 2)$-vertex connected set of size $2^{\mathcal{O}(k)}$. Our new tools yielded new FPT algorithms for Stable s-t Separator, Stable Odd Cycle Transversal, and Stable Multicut on general graphs, and for Stable Directed Feedback Vertex Set on d-degenerate graphs, resolving two problems left open by Marx et al. [Marx et al. 2013]. Observe that similar results will hold for a variant of these problems where instead of the solution being independent, one asks for a solution that induces an r-partite graph, for some fixed r. To get this, one can first find a k-independent set covering family and then guess/choose r sets in this family such that each partition of the r-partite solution is contained inside exactly one of the chosen sets. By doing so, we again reduce our problem to an annotated problem where one needs to find a solution which is contained in the union of the r chosen sets. One of the most natural direction to pursue further is to find more applications of our tools than given in the paper. Apart from this there are several natural questions that arise form our work.

(1) In the Stable Multicut problem we ask for a multicut that forms an independent set. Instead of requiring that the solution S is independent, we could require that it induces a graph that belongs to a hereditary graph class $\mathcal{G}$. Thus, corresponding to each hereditary graph class $\mathcal{G}$, we get the problem $\mathcal{G}$-Multicut. Is $\mathcal{G}$-Multicut FPT? Concretely, let $\mathcal{S}$ be the set of forests then is $\mathcal{S}$-Multicut FPT?
(2) Given a hereditary graph class $\mathcal{G}$, we can define the notion of k-$\mathcal{G}$ covering family, similar to k-independence covering family. Does there exist other hereditary families, apart from the family of independence sets, such that k-$\mathcal{G}$ covering family of FPT size exists?
(3) Observe that for all the problems whose non-stable version admit $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm on general graphs, such as s-t Separator and Odd Cycle Transversal, we get $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm for these problems on graphs of bounded degeneracy. As a corollary, we get $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm for these problems on planar graphs, graphs excluding some fixed graph H as minor or a topological minor and graphs of bounded degree. A natural question is whether these problems admit $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time algorithm on general graphs.

Acknowledgement. We thank the anonymous reviewers for their useful comments and suggestions.

REFERENCES

Béla Bollobás and Andrew Thomason. 1998. Proof of a Conjecture of Mader, Erdös and Hajnal on Topological Complete Subgraphs. *Eur. J. Comb.* 19, 8 (1998), 883–887.

Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. 2011. Multicut is FPT. In *STOC*. 459–468.

Jianer Chen, Yang Liu, and Songjian Lu. 2009. An Improved Parameterized Algorithm for the Minimum Node Multiway Cut Problem. *Algorithmica* 55, 1 (2009), 1–13.

Jianer Chen, Yang Liu, Songjian Lu, Barry O'sullivan, and Igor Razgon. 2008. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM (JACM)* 55, 5 (2008), 21.

Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. 2015. Directed Subset Feedback Vertex Set Is Fixed-Parameter Tractable. *ACM Transactions on Algorithms* 11, 4 (2015), 28.

Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. 2013. Fixed-Parameter Tractability of Directed Multiway Cut Parameterized by the Size of the Cutset. *SIAM J. Comput.* 42, 4 (2013), 1674–1696. https://doi.org/10.1137/12086217X

Hyeong-Ah Choi, Kazuo Nakajima, and Chong S. Rim. 1989. Graph Bipartization and via Minimization. *SIAM J. Discrete Math.* 2, 1 (1989), 38–47.

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

Marek Cygan, Lukasz Kowalik, and Marcin Pilipczuk. 2013a. Open problems from the update meeting on graph separation problems.

Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. 2013b. Subset Feedback Vertex Set Is Fixed-Parameter Tractable. *SIAM J. Discrete Math.* 27, 1 (2013), 290–309.

Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. 1994. The Complexity of Multiterminal Cuts. *Siam Journal on Computing* 23 (1994), 864–894. https://doi.org/10.1137/S0097539792225297

Erik D. Demaine, Gregory Gutin, Dániel Marx, and Ulrike Stege. 2007. 07281 Open Problems – Structure Theory and FPT Algorithmcs for Graphs, Digraphs and Hypergraphs. In *Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs, 08.07. - 13.07.2007.*

R. Diestel. 2000. *Graph Theory* (second ed., electronic ed.). Springer, Berlin.

Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2016. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM* 63, 4 (2016), 29:1–29:60. https://doi.org/10.1145/2886094

Lester R Ford and Delbert R Fulkerson. 1956. Maximal flow through a network. *Canadian journal of Mathematics* 8, 3 (1956), 399–404.

Michael L. Fredman, János Komlós, and Endre Szemerédi. 1984. Storing a Sparse Table with 0(1) Worst Case Access Time. *J. ACM* 31, 3 (1984), 538–544.

Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. 2013. Characterisations of Nowhere Dense Graphs (Invited Talk). In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, (LIPIcs)*, Vol. 24. 21–40.

Ken ichi Kawarabayashi and Bruce A. Reed. 2010. An (almost) Linear Time Algorithm for Odd Cycles Transversal. In *SODA*. 365–378.

Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. 2014. Linear-Time FPT Algorithms via Network Flow. In *SODA*. 1749–1761.

Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 85–103.

János Komlós and Endre Szemerédi. 1996. Topological cliques in graphs II. *Combinatorics, Probability & Computing* 5 (1996), 79–90.

Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Magnus Wahlström. 2015. Fixed-Parameter Tractability of Multicut in Directed Acyclic Graphs. *SIAM J. Discrete Math.* 29, 1 (2015), 122–144. https://doi.org/10.1137/120904202

Stefan Kratsch and Magnus Wahlström. 2012. Representative Sets and Irrelevant Vertices: New Tools for Kernelization. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*. IEEE Computer Society, 450–459. https://doi.org/10.1109/FOCS.2012.46

Daniel Lokshtanov and Dániel Marx. 2013. Clustering with local restrictions. *Inf. Comput.* 222 (2013), 278–292. https://doi.org/10.1016/j.ic.2012.10.016

Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. 2014. Faster Parameterized Algorithms Using Linear Programming. *ACM Trans. Algorithms* 11, 2 (2014), 15:1–15:31. https://doi.org/10.1145/2566616

Daniel Lokshtanov and M. S. Ramanujan. 2012. Parameterized Tractability of Multiway Cut with Parity Constraints. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*. 750–761. https://doi.org/10.1007/978-3-642-31594-7_63

Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. 2015. Linear Time Parameterized Algorithms for Subset Feedback Vertex Set. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*. 935–946. https://doi.org/10.1007/978-3-662-47672-7_76

Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. 2016a. A Linear Time Parameterized Algorithm for Directed Feedback Vertex Set. *CoRR* abs/1609.04347 (2016). http://arxiv.org/abs/1609.04347

Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. 2016b. A Linear Time Parameterized Algorithm for Node Unique Label Cover. *CoRR* abs/1604.08764 (2016). http://arxiv.org/abs/1604.08764

W. Mader. 1972. Existenzn-fach zusammenhängender Teilgraphen in Graphen genügend großer Kantendichte. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 37, 1 (1972), 86–97. https://doi.org/10.1007/BF02993903

Dániel Marx. 2006. Parameterized graph separation problems. *Theor. Comput. Sci.* 351, 3 (2006), 394–406. https://doi.org/10.1016/j.tcs.2005.10.007

Dáaniel Marx, Barry O'sullivan, and Igor Razgon. 2013. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms (TALG)* 9, 4 (2013), 30.

Dániel Marx and Igor Razgon. 2014. Fixed-Parameter Tractability of Multicut Parameterized by the Size of the Cutset. *SIAM J. Comput.* 43, 2 (2014), 355–388.

David W Matula and Leland L Beck. 1983. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)* 30, 3 (1983), 417–427.

Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. 2012. On parameterized independent feedback vertex set. *Theoretical Computer Science* 461 (2012), 65–75.

Jaroslav Nešetřil and Patrice Ossona de Mendez. 2008. Grad and classes with bounded expansion I. Decompositions. *European Journal of Combinatorics* 29, 3 (2008), 760–776.

Jaroslav Nešetřil and Patrice Ossona de Mendez. 2011. On nowhere dense graphs. *European Journal of Combinatorics* 32, 4 (2011), 600–617.

Jaroslav Nesetril and Patrice Ossona de Mendez. 2012. *Sparsity - Graphs, Structures, and Algorithms.* Algorithms and combinatorics, Vol. 28. Springer.

MS Ramanujan and Saket Saurabh. 2014. Linear time parameterized algorithms via skew-symmetric multicuts. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1739–1748.

Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. 2004. Finding odd cycle transversals. *Oper. Res. Lett.* 32, 4 (2004), 299–301.

Neil Robertson and Paul D. Seymour. 1995. Graph Minors. XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B* 63, 1 (1995), 65–110.

Mechthild Stoer and Frank Wagner. 1997. A simple min-cut algorithm. *Journal of the ACM (JACM)* 44, 4 (1997), 585–591.

Benny Sudakov. 2016. Graph Theory. *Lecture Notes* (2016). https://doi.org/education/bachelor/lectures/fs2016/math/graph_theory/graph_theory_notes.pdf