

Covering Small Independent Sets and Separators with Applications to Parameterized Algorithms

DANIEL LOKSHTANOV, University of California, USA

FAHAD PANOLAN, Department of Computer Science and Engineering, IIT Hyderabad, India

SAKET SAURABH, Institute of Mathematical Sciences, HBNI, India, University of Bergen, Norway

ROOHANI SHARMA, Institute of Mathematical Sciences, HBNI, India

MEIRAV ZEHAZI, 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 $\left(\binom{d+1}{k} \cdot k(d+1)\right)^{-1}$. The second is a new (deterministic) polynomial time graph sparsification procedure that given a graph G , a set $T = \{\{s_1, t_1\}, \{s_2, t_2\}, \dots, \{s_\ell, t_\ell\}\}$ of terminal pairs, and an integer k , returns an induced subgraph G^* 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 $2^{O(k)}$. In particular, G^* excludes a clique of size $2^{O(k)}$ 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: Independence covering family, stable multicut, stable s - t separator, stable OCT, parameterized algorithms

A preliminary version of this article has appeared in the proceedings of SODA 2018. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 819416), Pareto-Optimal Parameterized Algorithms (ERC Starting Grant 715744), Parameterized Approximation (ERC Starting Grant 306992), and Rigorous Theory of Preprocessing (ERC Advanced Investigator Grant 267959), and from the Norwegian Research Council via grant MULTIVAL. Saket Saurabh also acknowledges the support of Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.



Authors' addresses: D. Lokshtanov, University of California, Santa Barbara, USA; email: danielo@ucsb.edu; F. Panolan, Department of Computer Science and Engineering, IIT Hyderabad, Sangareddy, India; email: fahad@iith.ac.in; S. Saurabh, Institute of Mathematical Sciences, HBNI, Chennai, Tamil Nadu, India, University of Bergen, Bergen, Norway; email: saket@imsc.res.in; R. Sharma, Institute of Mathematical Sciences, HBNI, Chennai, Tamil Nadu, India; email: roohani@imsc.res.in; M. Zehavi, Ben-Gurion University, Beersheva, Israel; email: meiravze@bgu.ac.il.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.

1549-6325/2020/05-ART32 \$15.00

<https://doi.org/10.1145/3379698>

ACM Reference format:

Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. 2020. Covering Small Independent Sets and Separators with Applications to Parameterized Algorithms. *ACM Trans. Algorithms* 16, 3, Article 32 (May 2020), 31 pages.
<https://doi.org/10.1145/3379698>

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 $(\binom{k(d+1)}{k} \cdot k(d+1))^{-1}$. 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 $\sigma : V(G) \rightarrow \{1, \dots, n\}$ 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 linear¹ 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 $(\binom{k(d+1)}{k} \cdot k(d+1))^{-1}$.*

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 colored black with probability p and white with probability $1 - p$. The algorithm then constructs the set Y that contains every vertex v , such that v 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 $|Z| \leq kd$. 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

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

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

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].

¹The time is purely linear in terms of k and d , too.

- **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 $O(4^k \cdot k^{O(1)} \cdot (n + m))$ time algorithms of Iwata et al. [2014] and Ramanujan and Saurabh [2014] are the most relevant.
- **MULTICUT:** Here, the input is a graph G , a set $T = \{\{s_1, t_1\}, \{s_2, t_2\}, \dots, \{s_\ell, t_\ell\}\}$ 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 $i \leq \ell$, 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 three terminal pairs; that is, when $\ell = 3$ [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 $2^{O(k^3)} \cdot mn \log n$.
- **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 $O(k! \cdot 4^k \cdot k^5 \cdot (n + m))$.

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. [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 $2^{2^{k^{O(1)}}} \cdot (n + m) + O(3^k \cdot nm)$. Here, the $O(3^k \cdot nm)$ term in the running time comes from a direct invocation of the algorithm of Reed et al. [2004] for ODD CYCLE TRANSVERSAL. Furthermore, 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 Michał Pilipczuk at the update meeting on graph separation problems in 2013 [Cygan et al. 2013a].

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

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

Lemma 1.1 only applies to graphs of bounded degeneracy. Even though the class of graphs of bounded degeneracy is quite 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. [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 $2^{k^{O(1)}} \cdot (n + m)$, thus resolving, in the affirmative, one of the open problems of 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. [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 $T = \{s_1, t_1\}, \{s_2, t_2\}, \dots, \{s_\ell, t_\ell\}$ of terminal pairs and an integer k , returns an induced subgraph G^* of G and a subset T^* of T that have the following properties:*

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

We remark that excluding a $(k + 2)$ -connected set of size $O(64^k \cdot k^2)$ implies that G^* excludes a clique of size $O(64^k \cdot k^2)$ 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 [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 $2^{O(k)}$ 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. [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, which 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 $O\left(\binom{k(d+1)}{k}\right) \cdot kd \cdot \log n$ 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 $O\left(\binom{k(d+1)}{k}\right) \cdot kd \cdot \log n$. By direct applications of existing pseudo-random constructions (of $(n, (r, s))$ -cover free families), we show that given a graph G of degeneracy d and integer k one can construct a k -independence covering family of size not larger than $O\left(\binom{k(d+1)}{k}\right) \cdot kd \cdot \log n$ in time roughly proportionate to its size.

Additionally, we also show that for any *nowhere dense* graph class [Nešetřil and Ossana de Mendez 2008, 2011], there exists a function f such that given an n -vertex graph from this graph class, any real ϵ and any positive integer k , one can construct a k -independence covering family for this graph of size $f(k, \epsilon) \cdot n^\epsilon$. 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 $|W| \leq 64^{k+2} \cdot (k+2)^2$.

Degree 1 Terminals Assumption. 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 $s_i^1, s_i^2, \dots, s_i^{k+2}$ and make them adjacent to s_i , and $k+2$ new degree 1 vertices $t_i^1, t_i^2, \dots, t_i^{k+2}$ 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 $\{\{s_i^j, t_i^j\} : 1 \leq j \leq k+2\}$ 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. 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 $|S| \leq k+1$, $W \setminus S$ 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 $W \setminus S$ lie 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 $|S| \leq k+1$, $v \in S$, 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 $v \in W$ that is provably not essential.

The algorithm that searches for a non-essential vertex v crucially exploits *important separators*, defined by 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 $|S_1| \leq |S_2|$ and $R_G(A, S_2) \subsetneq R_G(A, S_1)$. 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 [2006] and the textbook by 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. As pointed out by an anonymous reviewer, in both the cases, the essence of the arguments is to set up the stage for the application of the anti-isolation lemma of Marx [2011], which appeared in Pilipczuk and Wahlström [2018], to mark relevant vertices and relevant terminal pairs, respectively. 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 , then 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 is not 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 $A = \{s_1, s_2, \dots, s_{\zeta/2}\}$ 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 $B = \{t_1, t_2, \dots, t_{\zeta/2}\}$. 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 $W \setminus S$ 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 $|\hat{B}| \leq k+2$. 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^* 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 $t_i \in S^*$. 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. We would like to remark here (as pointed out by an anonymous reviewer) that this process is similar in principle to the anti-isolation lemma of Marx [2011] (which also appeared in Pilipczuk and Wahlström [2018]).

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 not 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 [1972] (see also Diestel [2000] and lecture notes of 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 article. 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. [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 article 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 $\{1, \dots, n\}$. 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 $f : D \rightarrow R$, $X \subseteq D$ and $Y \subseteq R$, we denote $f(X) = \{f(x) : x \in X\}$ and $f^{-1}(Y) = \{d : f(d) \in Y\}$. The following fact follows from Stirling's approximation:

FACT 2.1 ([CORMEN ET AL. 2009]). *For all positive integers $n, k, k \leq n$,*

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

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 its arc set in case of digraphs, and $E(G)$ denotes its 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 and 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 $deg_G(v)$, is equal to the number of neighbors of v in G ; that is, $deg_G(v) = |N_G(v)|$. 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 $X \subseteq V(D)$, $N_D^+(X) = \{u : u \in V(D) \setminus X \text{ and there exists } v \in X \text{ such that } vu \in A(D)\}$ and $N_D^-(X) = \{u : u \in V(D) \setminus X \text{ and there exists } v \in X \text{ such that } uv \in A(D)\}$. For a graph G , $\text{tw}(G)$ 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 $v \in V(H)$ such that $deg_H(v) \leq d$. The degeneracy of a graph G , denoted by $\text{degeneracy}(G)$, 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 , then 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 $\sigma : V(G) \rightarrow [|V(G)|]$, such that σ is a bijection and, for any $v \in V(G)$, $|N_G(v) \cap \{u : \sigma(u) > \sigma(v)\}| \leq d$. 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 $N_{G,\sigma}^f(v)$. 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 $O(n + m)$.*

Graph Separators. For a (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 $s, t \in V(G)$ 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 $G - C$. 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 *mincut* of G if Y is the smallest set of vertices such that $G - Y$ 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-Fulkerson algorithm, Proposition 2.2 follows:

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

The following proposition follows directly from the standard reduction that reduces finding minimum vertex separators to finding minimum edge cuts in directed graphs and the result about the later in Hao and Orlin [1992].

PROPOSITION 2.3 ([HAO AND ORLIN 1992]). *A mincut of a (di)graph G can be found in time $O(mn \log n)$.*

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.

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

- 1 Construct a d -degeneracy sequence σ of G , using Proposition 2.1.
 - 2 Set $p = \frac{1}{d+1}$. Independently color each vertex $v \in V(G)$ black with probability p and white with probability $(1 - p)$.
 - 3 Let B and W be the set of vertices colored black and white, respectively.
 - 4 $Z := \{v \in B \mid N_{G, \sigma}^f(v) \cap B = \emptyset\}$.
 - 5 **return** Z
-

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

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) $\gamma = t \cdot 2k \ln n$ times. Let Z_1, \dots, Z_γ 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, $|\mathcal{F}(G, k)| \leq t \cdot 2k \ln n = \binom{k(d+1)}{k} \cdot 2k^2(d+1) \cdot \ln n$. 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 $Z \in \mathcal{F}(G, k)$, $\Pr[X \subseteq Z] \geq \frac{1}{t}$. Thus, the probability that there does not exist a set $Z \in \mathcal{F}(G, k)$ such that $X \subseteq Z$ is at most $(1 - \frac{1}{t})^{|\mathcal{F}(G, k)|} \leq e^{-2k \ln n} = n^{-2k}$. The last inequality follows from a well-known fact that $(1 - a) \leq e^{-a}$ for any $a \geq 0$. 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 that is not a subset of any set in $\mathcal{F}(G, k)$ is upper bounded by $n^{-2k} \cdot n^k = n^{-k} \leq 1/n$. This implies that $\mathcal{F}(G, k)$ is an independence covering family for (G, k) with probability at least $1 - \frac{1}{n}$. \square

Remark 3.1. From Fact 2.1 and the fact that the number of edges in an n -vertex d -degenerate graph is at most dn , the algorithm of Lemma 3.1 runs in time $2^{O(k \log d)} \cdot n \log n$ and outputs a k -independence covering family of size $2^{O(k \log d)} \cdot \log n$.

Deterministic Construction. The deterministic algorithm that we give is obtained from the randomized algorithm presented in Lemma 3.1 by using the $(n, (r, s))$ -cover free family [Bshouty and Gabizon 2017]. The deterministic construction basically replaces the random coloring of the vertices in Line 2 of Algorithm 1 by a coloring defined by a bit string in the $(n, (r, s))$ -cover free family. In the following, we first define the $(n, (r, s))$ -cover free family and state Proposition 3.1 (an algorithm to construct an $(n, (r, s))$ -cover free family of “small” size), which is followed by our deterministic algorithm (Lemma 3.2).

Definition 3.2 (($n, (r, s)$)-cover free family [Bshouty and Gabizon 2017]). Fix positive integers r, s, n with $r, s \leq n$ and let $p := r + s$. An $(n, (r, s))$ -cover free family is a set $\mathcal{F} \subseteq \{0, 1\}^n$ such that for every $1 \leq i_1 < i_2 < \dots < i_p \leq n$ and every $J \subseteq [p]$ of size r , there exists $\mathbf{a} \in \mathcal{F}$ such that $\mathbf{a}_{i_j} = 1$ for all $j \in J$ and $\mathbf{a}_{i_k} = 0$ for all $k \notin J$. Here, \mathbf{a}_{i_j} denotes the i_j th bit of the bit vector \mathbf{a} .

In the following, for any positive integers r, s , and $p = r + s$, the function $N(r, s)$ is defined as
$$N(r, s) = \frac{\binom{p}{r}}{\log \binom{p}{r}}.$$

PROPOSITION 3.1 (THEOREM 1, [BSHOUTY AND GABIZON 2017]). *Fix any integers $r < s < p$ with $p = r + s$. There is an $(n, (r, s))$ -cover free family of size $N(r, s)^{1+o(1)} \cdot \log n$ that can be constructed in time $N(r, s)^{1+o(1)} \cdot n \log n$.*

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 $N(k, kd)^{1+o(1)}(n + m) \log n$ and outputs a k -independence covering family for (G, k) of size at most $N(k, kd)^{1+o(1)} \cdot \log n$.*

PROOF. Let $n = |V(G)|$. Without loss of generality, let $n \geq k(d + 1)$, as otherwise the lemma follows trivially. Let us rename the vertex set of the graph to take indices from $[n]$, where $n = |V(G)|$. Let \mathcal{F} be the $(n, (r, s))$ -cover free family constructed using Proposition 3.1 for $r = k$ and $s = kd$. For each $\mathbf{a} \in \mathcal{F}$, we run Algorithm 1, where Line 2 is replaced as follows: We color the vertex i black if $\mathbf{a}_i = 1$, and white otherwise. More precisely, we run Algorithm 1 for each $\mathbf{a} \in \mathcal{F}$, 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 $Y = \cup_{v \in X} N_{G, \sigma}^f(v)$. Since X is independent, $X \cap Y = \emptyset$. Furthermore,

since σ is a d -degeneracy sequence and $|X| \leq k$, we have that $|Y| \leq kd$. If $|X| < k$ (or $|Y| < kd$), then let X' (respectively, Y') be a some superset of X (respectively, Y) such that $X' \cap Y' = \emptyset$ and $|X'| = k$, $|Y'| = kd$. Since $n \geq k(d+1)$ such sets X', Y' exist. Let $X' \cup Y' = \{i_1, i_2, \dots, i_p\}$, where $p = k(d+1)$ and let $J \subset [p]$ be such that $J = \{j : i_j \in X', j \in [p]\}$. By the definition of $(n, (k, kd))$ -cover free family, there is a bit vector $\mathbf{a} \in \mathcal{F}$ such that $\mathbf{a}_{i_j} = 1$ when $i_j \in X'$ and $\mathbf{a}_{i_j} = 0$ when $i_j \in Y'$. Consider the run of Algorithm 1 for the bit vector \mathbf{a} . 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. \square

Remark 3.2. From Fact 2.1 and the fact that the number of edges in an n -vertex d -degenerate graph is at most dn , the algorithm of Lemma 3.2 runs in time $2^{O(k \log d)} \cdot n \log n$ and outputs a k -independence covering family for (G, k) of size $2^{O(k \log d)} \cdot \log n$.

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 Corollary 3.1.

COROLLARY 3.1. *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 is an algorithm that given $d, k \in \mathbb{N}$, G , and S , run in time $2^{|S|} \cdot 2^{O(k \log d)} \cdot (n+m) \log n$ and outputs an independence covering family for (G, k) of size at most $2^{|S|} \cdot 2^{O(k \log d)} \cdot \log n$.*

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 $|\mathcal{F}(G, k)| \leq |\mathcal{F}'| \cdot 2^{|S|}$. 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 Lemma 3.2 and Remark 3.2. \square

Remark 3.3³. An alternate independence covering family for the situation in Corollary 3.1 can be obtained directly from Lemma 3.2 by observing that the input graph has degeneracy at most $d + |S|$. This procedure gives an independence covering family whose size (in terms of the dependence on $|S|$) has a factor of $|S|^{O(k)}$ in contrast to $2^{|S|}$ in Corollary 3.1. Thus, the result of Corollary 3.1 is relevant only when $d \ll |S| \ll k$.

3.2 Nowhere Dense Graphs

In this section, we show that for any *nowhere dense* graph class [Nešetřil and Ossana de Mendez 2008, 2011], there exists a function f such that given an n -vertex graph from this graph class, any real ϵ , and any positive integer k , one can construct a k -independence covering family for this graph of size $f(k, \epsilon) \cdot n^\epsilon$. The class of graphs that is nowhere dense is a common generalization of

³Pointed out to us by an anonymous reviewer.

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 Ossona de Mendez [2011], Figure 3). Also, they are incomparable to the class of bounded degeneracy graphs [Brandstadt et al. 1999; Nešetřil and Ossona de Mendez 2009]. To define nowhere denseness, we need several new definitions.

Definition 3.3 (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 $\psi : V(M) \rightarrow \{V_1, \dots, V_{|M|}\}$ 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 $G \nabla r$. Similarly, the set of all r -shallow minors of all the members of a graph class \mathcal{G} is denoted by $\mathcal{G} \nabla r = \cup_{G \in \mathcal{G}} (G \nabla r)$.

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

Definition 3.4 (Nowhere dense). A graph class \mathcal{G} is *nowhere dense* if there exists a function $f_\omega : \mathbb{N} \rightarrow \mathbb{N}$ such that for all r we have that $\omega(\mathcal{G} \nabla r) \leq f_\omega(r)$.

We refer the readers to the book by Nešetřil and Ossona de Mendez [2012] for a detailed exposition of nowhere dense classes of graphs, their alternate characterizations, and several properties of them. See also Grohe et al. [2013]. We rely on the following result that bounds the degeneracy of any class of graphs that is nowhere dense to give a construction for independence covering family for such graph classes.

PROPOSITION 3.2 (COROLLARY 2.6, [GROHE ET AL. 2013]). *Let \mathcal{G} be a class of graphs that is nowhere dense. There exists a function f such that for every real $\epsilon > 0$ and every $G \in \mathcal{G}$, the degeneracy of G is $f(\epsilon) \cdot n^\epsilon$.*

We now give the construction of independence covering family for the class of graphs that are nowhere dense.

LEMMA 3.3. *Let \mathcal{G} be a class of graphs that is nowhere dense. Then there exists a function g and a deterministic algorithm that given any $k \in \mathbb{N}$, $\delta \in \mathbb{R}$ and $G \in \mathcal{G}$, computes in time $g(\delta, k) \cdot n^{1+\delta} \log n$ a k -independence covering family for (G, k) of size at most $g(\delta, k) \cdot n^\delta \log n$.*

PROOF. From Remark 3.2, an independence covering family for G of size at most $2^{ck \log d} \log n$ can be computed in time $2^{ck \log d} n \log n$, where d is the degeneracy of G and c is an absolute constant. Set $\epsilon = \frac{\delta}{ck}$. Since $G \in \mathcal{G}$ and \mathcal{G} is a class of graphs that is nowhere dense, from Proposition 3.2, $d = f(\epsilon) \cdot n^\epsilon$ for some function f . Thus, we obtain an independence covering family for G of size $(f(\frac{\delta}{ck}))^{ck} \cdot n^\delta \log n$ in time $(f(\frac{\delta}{ck}))^{ck} \cdot n^{1+\delta} \log n$. Setting $g(\delta, k) = (f(\frac{\delta}{ck}))^{ck}$, we prove the lemma. \square

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, \dots, C_k be the disjoint cliques that compose G . Let $\mathcal{F}(G, k)$ be a k -independence covering family for (G, k) . Then, we claim that $|\mathcal{F}(G, k)| \geq (\frac{n}{k})^k$. Consider the family \mathcal{I} of independent sets of G of size at most k defined as $\mathcal{I} = \{\{v_1, \dots, v_k\} : \forall i \in [k], v_i \in C_i\}$. Note that $|\mathcal{I}| = (\frac{n}{k})^k$. 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 $|\mathcal{F}(G, k)| \geq (\frac{n}{k})^k$. 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 3.5 (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 $\frac{k}{2}$ 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 \mathcal{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-independent 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-independent set* in G if for any $u, v \in V(G)$, $d_G(u, v) > r$. An independent set in G is a 1-independent set in G .

Definition 3.6 (r-independent 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 an *r-independent covering family* for (G, k) if for all $Y \in \mathcal{S}$, Y is an r -independent set in G and for any $X \subseteq V(G)$ of size at most k such that X is an r -independent 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, \dots, C_k be the components of G . Define $\mathcal{I} = \{\{v_1, \dots, v_k\} : \forall i \in [k], v_i \in C_i\}$. Clearly, each set in \mathcal{I} is a r -independent set for any $r \in \mathbb{N}$. Moreover, the union of any two distinct sets in \mathcal{I} is not a 2-independent set. This implies that the cardinality of any r -independent covering family for (G, k) is at least $|\mathcal{I}| = (\frac{n}{k})^k$ for any $r \geq 2$.

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 3.7 (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 $Y \in \mathcal{A}$, $G[Y]$ 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 $K_{2, \frac{3n}{k}-2}$. The degeneracy of G is 2. Without loss

of generality, assume that $\frac{3n}{k}$ is strictly more than 2. Let $H_1, \dots, H_{\frac{k}{3}}$ 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 $\mathcal{I} = \{L_1 \cup \dots \cup L_{\frac{k}{3}} \cup \{v_1, \dots, v_{\frac{k}{3}}\} \mid v_i \in R_i\}$. Each set in \mathcal{I} induces a collection of induced paths on three vertices (P_3). Also, the union of any two sets in \mathcal{I} contains a cycle on four vertices and, hence, not acyclic. This implies that the cardinality of any acyclic set covering family for (G, k) is at least $|\mathcal{I}| = (\frac{3n}{k} - 2)^{\frac{k}{3}}$.

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 and 3.2. 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.

We begin by defining a general algorithmic framework that will be applicable to each of the algorithms in this section. To this end, we define Π -VERTEX DELETION, ANNOTATED Π -VERTEX DELETION, and STABLE Π -VERTEX DELETION problems, for any graph class Π .

Π -VERTEX DELETION

Parameter: k

Input: An instance \mathcal{I} of a graph problem containing a graph G , an integer k

Question: Does there exist $S \subseteq V(G)$, such that $|S| \leq k$ and $G - S \in \Pi$?

ANNOTATED Π -VERTEX DELETION

Parameter: k

Input: An instance \mathcal{I} of a graph problem containing a graph G , a subset $Y \subseteq V(G)$, an integer k

Question: Does there exist $S \subseteq Y$, such that $|S| \leq k$ and $G - S \in \Pi$?

STABLE Π -VERTEX DELETION

Parameter: k

Input: An instance \mathcal{I} of a graph problem containing a graph G , an integer k

Question: Does there exist $S \subseteq V(G)$, such that $|S| \leq k$, S is an independent set and $G - S \in \Pi$?

Using our constructions of the independence covering families, the following lemma describes a procedure to design FPT algorithms for STABLE Π -VERTEX DELETION problems using FPT algorithms for ANNOTATED Π -VERTEX DELETION, for graphs of bounded degeneracy. In the following, for any positive integers r, s , and $p = r + s$, $N(r, s) = \frac{p \binom{p}{r}}{\log \binom{p}{r}}$.

LEMMA 4.1. *If there is an algorithm that solves ANNOTATED Π -VERTEX DELETION on a d -degenerate graph on n vertices in time $T(d, n)$, then STABLE Π -VERTEX DELETION on d -degenerate graphs can be solved by*

- (1) *a randomized algorithm with worst case running time $(T(d, n) + (n + m)) \cdot \binom{k(d+1)}{k} \cdot k^2(1 + d)$ that always outputs correctly if the instance is a No instance and makes an error with probability at most $1 - 1/e$ if it is Yes instance; and*
- (2) *a deterministic algorithm that runs in time $(T(d, n) + (n + m)) \cdot N(k, kd)^{1+o(1)} \log n$.*

PROOF. We first begin by describing our randomized algorithm.⁴ Let (\mathcal{I}, k) be an instance of STABLE Π -VERTEX DELETION and let G be the graph of the instance \mathcal{I} . Our algorithm runs the following two step procedure $\binom{k(1+d)}{k} \cdot k(1+d)$ many times:

⁴To 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 $\mathcal{F}(G, k)$ beforehand using multiple rounds of Algorithm 1.

- (1) Run Algorithm 1 on (G, k) and let Z be its output.
- (2) Run the algorithm of ANNOTATED Π -VERTEX DELETION on the instance (\mathcal{I}, k, Z) .

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 $(1-p)^{1/p} \leq 1/e$. 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, (\mathcal{I}, k, Z) is a Yes instance of ANNOTATED Π -VERTEX DELETION, 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 Lemma 1.1 and the running time for ANNOTATED Π -VERTEX DELETION.

For our deterministic algorithm, the algorithm first computes a k -independence covering family for (G, k) , $\mathcal{F}(G, k)$ using the algorithm of Lemma 3.2. For each $Z \in \mathcal{F}(G, k)$, it then solves the instance (\mathcal{I}, k, Z) of ANNOTATED Π -VERTEX DELETION. If the algorithm of ANNOTATED Π -VERTEX DELETION returns Yes on either of the instances, then our algorithm reports Yes, otherwise it reports No. The correctness of the algorithm follows from the definition of independent set covering family and discussion done in the above paragraph. The running time of the algorithm follows from Lemma 3.2 and the running time to solve ANNOTATED Π -VERTEX DELETION. \square

The rest of the section focuses on four Π -VERTEX DELETION problems viz. s - t -SEPARATOR, ODD CYCLE TRANSVERSAL (OCT), DIRECTED FEEDBACK VERTEX SET (DFVS), and MULTICUT. In s - t -Separator, the instance \mathcal{I} contains a graph G and $s, t \in V(G)$, and Π is the class of graphs that contain the vertices s, t and, s and t belong to different connected components. In OCT, the instance \mathcal{I} contains a graph G , and Π is the collection of all bipartite graphs. In DFVS, the instance \mathcal{I} contains a directed graph⁵ D , and Π is the collection of all acyclic directed graphs. In MULTICUT, the instance \mathcal{I} contains a graph G and a set $T = \{(s_i, t_i) : i \in [p]\}$ of terminal pairs, and Π is the collection of graphs where there is no path from s_i to t_i for each $i \in [p]$.

Using the framework of Lemma 4.1 and by designing simple algorithms for the ANNOTATED Π -VERTEX DELETION problems corresponding to the above-mentioned problems from the algorithms of the corresponding Π -VERTEX DELETION problems, we get the following theorem:

THEOREM 2. *There is a randomized algorithm with one-sided error probability $1/e$ and a deterministic algorithm for*

- (1) STABLE s - t SEPARATOR (SSTS) and STABLE ODD CYCLE TRANSVERSAL (SOCT) on d -degenerate graphs that run in time $2^{O(k \log d)} \cdot n$ and $2^{O(k \log d)} \cdot n \log n$, respectively,
- (2) STABLE DIRECTED FEEDBACK VERTEX SET (SDFVS) on d -degenerate graphs that run in time $(k+1)! \cdot 2^{O(k \log d)} \cdot n$ and $(k+1)! \cdot 2^{O(k \log d)} \cdot n \log n$, respectively, and
- (3) STABLE MULTICUT on d -degenerate graphs that run in time $2^{O(k^3 + k \log d)} \cdot mn \log^2 n$.⁶

⁵In this article, we abuse the notation a little and whenever we refer to the degeneracy of a directed graph, we mean the degeneracy of its underlying undirected graph.

⁶The randomized algorithm for STABLE MULTICUT does not give any better running time than the deterministic one, so for the sake of soundness of the sentence, we may assume that the randomized algorithm is the same as the deterministic algorithm.

To prove Theorem 2, it is enough to design appropriate algorithms for the annotated versions of these problems, which we do below. Henceforth, an instance of ANNOTATED s - t -SEPARATOR, ANNOTATED OCT (AOCT), ANNOTATED DFVS (ADFVS), and ANNOTATED MULTICUT is (G, s, t, Y, k) , (G, Y, k) , (D, Y, k) , and (G, T, Y, k) , respectively.

LEMMA 4.2. *ASTS can be solved in time $O(k \cdot (n + m))$.*

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. \square

We will need the following result about OCT:

PROPOSITION 4.1 ([RAMANUJAN AND SAURABH 2014]). *OCT can be solved in time $O(4^k \cdot k^4 \cdot (n + m))$.*

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

LEMMA 4.3. *AOCT can be solved in time $O(4^k \cdot k^6 \cdot (n + m))$.*

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, \dots, v_{k+1} with same neighborhood as v ; that is, the neighborhood of v_1, \dots, v_{k+1} are same in the resulting graph (see Figure 3 for an illustration). Let G' be the resulting graph. Then any minimal odd cycle transversal that contains a vertex from $\{v_1, \dots, v_{k+1}\}$ will also contain all the vertices in $\{v_1, \dots, v_{k+1}\}$. 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

We need to use the following known algorithm for DFVS:

LEMMA 4.4 ([LOKSHTANOV ET AL. 2016A]). *DFVS can be solved in time $O((k + 1)! \cdot 4^k \cdot k^5 \cdot (n + m))$.*

LEMMA 4.5. *ADFVS can be solved in $O((k + 1)! \cdot 4^k \cdot k^7 \cdot (n + m))$ time.*

PROOF SKETCH. Construct G' as in Lemma 4.3; 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 4.4 on (G', k) . The proof of correctness of this algorithm is similar in arguments to the proof of Lemma 4.3. \square

Next, we state an algorithmic result for MULTICUT that is used by our algorithm.

LEMMA 4.6 ([LOKSHTANOV ET AL. 2016A; MARX 2006]). *MULTICUT can be solved in $2^{O(k^3)} \cdot mn \log n$ time.*

LEMMA 4.7. *ANNOTATED MULTICUT can be solved in time $2^{O(k^3)} \cdot mn \log n$.*

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 replacing each $v \in V(G) \setminus Y$ with $k + 1$ vertices v_1, \dots, v_{k+1} with same neighborhood as v . That is, the neighborhood of v_1, \dots, v_{k+1} are same in the resulting graph G' . We call the set of vertices that is 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 $\{s, t\} \subseteq Y$, then we add $\{s, t\}$ to T' . Suppose $\{s, t\} \in T$ and $\{s, t\} \cap Y = \{t\}$. Let s_1, \dots, s_{k+1} be the block for s in G' . We add $\{s_1, t\}, \dots, \{s_{k+1}, t\}$ to T' . Suppose

$\{s, t\} \in T$ and $\{s, t\} \subseteq V(G) \setminus Y$. Let s_1, \dots, s_{k+1} and t_1, \dots, 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, \dots, v_{k+1} in G' is the same as that of v in G and $|C'| \leq k$, $C' \cap \{v_1, \dots, v_{k+1}\} = \emptyset$. 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 that 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 4.6. This completes the proof sketch of the lemma. \square

The proof of Theorem 2 follows from Lemmas 4.1, 4.2, 4.3, 4.5, and 4.7 and the fact that the number of edges in an n -vertex d -degenerate graph is at most dn .

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 Theorem 2. We begin by stating the Treewidth Reduction Theorem.

THEOREM 3 (TREWIDTH 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 $2^{(k+|T|)^{O(1)}} \cdot (n+m)$:*

- (1) $C \cup T \subseteq V(G^*)$,
- (2) for every $s, t \in T$, a set $K \subseteq V(G^*)$ with $|K| \leq k$ 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 $2^{(k+|T|)^{O(1)}}$, and
- (4) $G^*[C \cup T]$ is isomorphic to $G[C \cup T]$.

We remark here that Theorem 2.15 in Marx et al. [2013] does not state the explicit dependence on k and $|T|$ in the running time of the algorithm and the treewidth of G^* .

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 3) on $G, T = \{s, t\}$ and k to obtain a graph G^* with treewidth upper bounded by $2^{k^{O(1)}}$. We then show that for SSTS, it is enough to work with this new graph G^* . By conditions 2 and 4, to find a minimal independent s - t -separator in G , it is enough to find a minimal independent s - t -separator in G^* . Since degeneracy of a graph is at most its treewidth, we know that the degeneracy of G^* is at most $2^{k^{O(1)}}$ and, hence, we apply Theorem 2 to get a solution of SSTS on (G, k) . That is, we get the following theorem:

THEOREM 4. *There is a randomized algorithm that solves SSTS in time $2^{k^{O(1)}}(n+m)$ with success probability at least $1 - \frac{1}{e}$. There is a deterministic algorithm that solves SSTS in time $2^{k^{O(1)}}(n+m) \log n$.*

Stable Odd Cycle Transversal. By using Theorem 4 and Proposition 4.1, we get a $2^{k^{O(1)}}(n+m)$ time (FPT linear time) algorithm for SOCT. Towards that, in Theorem 4.2 of Marx et al. [2013], we replace the algorithm of Kawarabayashi and Reed [2010] with Proposition 4.1 and the algorithm for SSTS with Theorem 4. 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 5. *There is a randomized algorithm that solves SOCT in time $2^{k^{O(1)}}(n+m)$ with success probability at least $1 - \frac{1}{e}$. There is a deterministic algorithm that solves OCT in time $2^{k^{O(1)}}(n+m) \log n$.*

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 a No instance of SOCT. Henceforth, we can assume that such a set S_0 exists. Next, we 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), colored with the first color—say, black—or colored with the second color—say, white. For a particular branch, let R be the vertices of S_0 to be removed (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 4.

Towards the run time analysis, we run the algorithm of Proposition 4.1 once, which takes time $2^{O(k)}(m+n)$. Then, we apply Theorem 4 at most 3^k times. Thus, we get the required running time. \square

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^* of G and a subset $T^* \subseteq T$ 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 $S \subseteq V(G^*)$ and it is a minimal multicut of T^* in G^* . Second, G^* 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 [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 a T - r -digraph pair cut if for every terminal pair $\{s, t\} \in T$, S is an s - r -separator or a t - r -separator.⁷

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 a T - r -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 T - r -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 T - r -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 $O(|T| \cdot n(n^{\frac{2}{3}} + m))$, and outputs a set $R \subseteq N_D^-(r)$ of size at most $64^{k+1}(k + 1)^2$ that contains all relevant vertices to (D, T, r, k) in $N_D^-(r)$.⁸*

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 $(T \setminus \{\{s, t\}\})$ - r -digraph pair cut in D of size at most k is also a minimal T - r -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.

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

⁷The definition of digraph pair cut used here is same as that of Kratsch and Wahlström [2012], where we reverse the directions of the arcs of the graph.

⁸In other words, the vertices in $N_D^-(r) \setminus R$ are irrelevant to (D, T, r, k) .

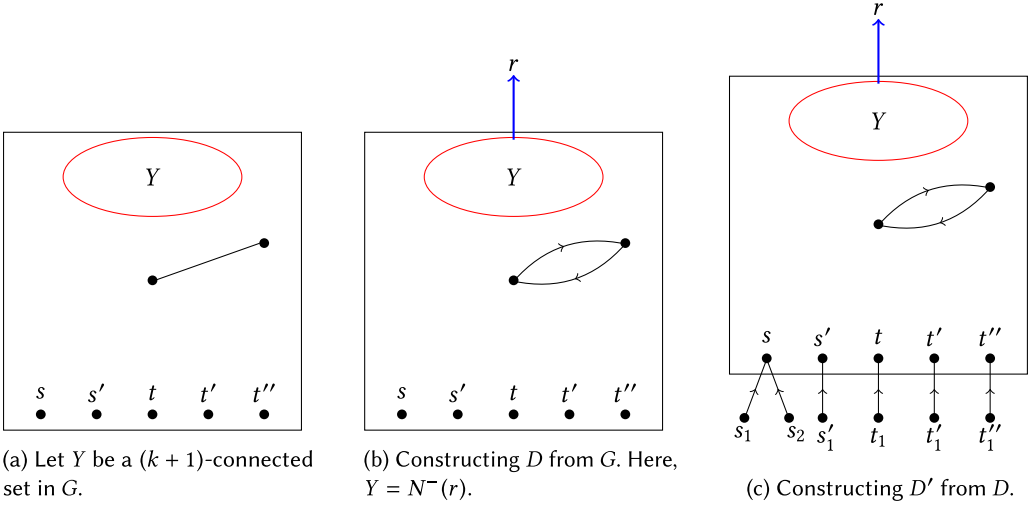


Fig. 1. The graphs G, D , and D' are displayed in left-to-right order, $T = \{\{s, t\}, \{s, t''\}, \{s', t'\}\}$ and $T' = \{\{s_1, t_1\}, \{s_2, t_1''\}, \{s_1', t_1'\}\}$.

S dominates an X - Y -separator S' if $|S| \leq |S'|$ and $R_D(X, S') \subsetneq R_D(X, S)$. 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 Y - r -separator* refers to an important Y - $N_D^+(r)$ -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 $|N_D^-(r)| \leq 64^{k+1}(k+1)^2$, 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 T - r -digraph pair cut in D is also a minimal T' - r -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 pairs in the set T' are pairwise disjoint (see Figures 1(b) and 1(c) 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.

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  $|Z| \leq 16^k \cdot 64(k+1)$  then
6   For each  $z \in Z$ , compute all important  $z$ - $r$ -separators of size at most  $k$ .
7   Mark all the vertices in  $N_{D'}^-(r)$  that 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  $\{s', t'\} \in T'$ ,  $|A \cap \{s', t'\}| \leq 1$ .
12  Let  $B = \{w \mid \text{there exists } w' \in A \text{ such that } \{w, w'\} \in T'\}$ . That is,  $B$  is the set of vertices that is 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  $B$  that 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.

```

The correctness of Algorithm 2 as exhibited in Lemma 6.3 is essentially based on creating a setup that allows the applicability of the anti-isolation lemma of Marx [2011], which appeared at Pilipczuk and Wahlström [2018], to find and mark relevant vertices and relevant terminal pairs.

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 $|Z| \cdot 4^k k + |Z| \leq 64^{k+1}(k+1)^2$. 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: $|Z| \leq 16^k \cdot 64(k+1)$. 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 T' - r -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 T' - r -digraph pair cut, S' is not a T' - r -digraph pair cut. Since S is a T' - r -digraph pair cut and S' is not a T' - r -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 $v \in Z$, then we are done. Thus, from now on, assume that $v \notin Z$. \square

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 pass 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 $R_{D'}(t, S)$. \square

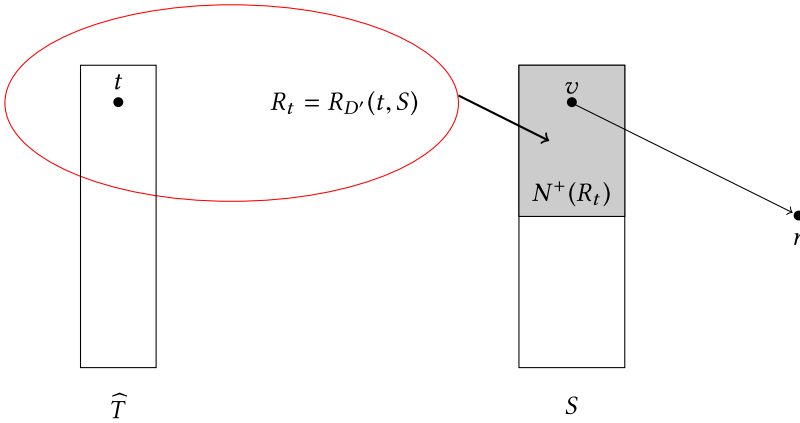


Fig. 2. Here, the ellipse contains the set of vertices reachable from t in $D' - S$, denoted by R_t . The rectangle colored grey represents $N^+(R_t)$, which includes v .

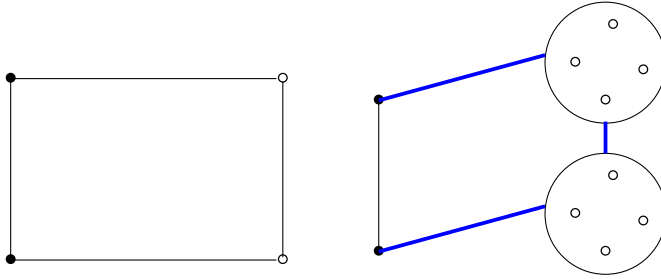


Fig. 3. The graph at the right-hand side is obtained by the reduction on the graph at the left-hand side, where $k = 3$ and Y is the set of black-colored vertices. Thick lines represent all possible edges between two sets of vertices.

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 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 2 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 $(T' \setminus \{\{q, q'\}\})$ - r -digraph pair cut of size at most k . We need to show that S is also a T' - r -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, $|\widehat{T}| \geq |Z|$. 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 be the set of vertices in B such that r is reachable from each vertex in B_r in the graph $D' - S$; 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 $|S| \leq k$, we have $|A_r| \geq |A| - (k+1)$. Since S is a $(T' \setminus \{\{q, q'\}\})$ - r -digraph pair cut, the vertices in B that are paired with a vertex in A_r are not reachable from r in $\overleftarrow{D'} - S$. This implies that $|B_r| \leq k+1$. 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 T' - r -digraph pair cut, because S is a $(T' \setminus \{\{q, q'\}\})$ - r -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 that dominate Q' will contain q . This 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 Observation 6.1 and induction hypothesis, Algorithm 2 will output the required set.

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

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 $O(mn)$. By Lemma 6.2, the time required to enumerate important separators in Lines 6 and 13 is bounded by $O(4^{2k} \cdot k \cdot (n+m))$. The time required to compute \mathcal{P} in Line 13 is $O(mn)$ by Proposition 2.2. Thus, the total running time of Algorithm 2 is $O(|T'| (mn + 4^{2k} \cdot k \cdot (n+m)))$. 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. \square

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 $T = \{\{s_1, t_1\}, \dots, \{s_\ell, t_\ell\}\}$, and an integer k , there is a polynomial time algorithm that finds a pair (G^*, T^*) , where G^* is an induced subgraph of G such that it has no $(k+2)$ -connected sets of size $2^{O(k)}$ and $T^* \subseteq T$ 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 if S is a subset of $V(G^*)$ and S is a minimal multicut of T^* in G^* . 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 that runs in time $O(|T| \cdot n^2(n^{\frac{2}{3}} + m) + n^5 \log n)$ and outputs an induced subgraph G^* of G and a subset $T^* \subseteq T$ such that

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

The proof of Lemma 6.5 requires some auxiliary lemmas that we discuss below. Recall the definition of the problem MULTICUT from Section 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 arcs with the same endpoints and opposite orientations. Then, 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 1(a) and (b). 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 a T - r -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 $|C| \leq k$, 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 T - r -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 a T - r -digraph pair cut in D , there exists $C' \subseteq C$ such that C' is a minimal T - r -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 a T - r -digraph pair cut in D , which is a contradiction. \square

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 that given a graph G and $k, d \in \mathbb{N}$, $k \leq d$, runs in time $O(n^4 \log n)$ 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 [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] and 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 that, for any $d \in \mathbb{N} \setminus \{0\}$, given a graph G with degeneracy at least $4d$, runs in time $O(n^2 m \log n)$ and outputs a $(d + 1)$ -connected subgraph of G .*

PROOF. The algorithm first constructs a subgraph H of G that 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 $O(mn)$. \square

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

PROOF. Since minimum degree of H is at least $4d$, clearly $|V(H)| \geq 4d + 1 \geq 2d + 1$. Also, since $\sum_{v \in V(H)} \deg_G(v) = 2|E(H)|$ and for all $v \in V(H)$ $\deg_G(v) \geq 4d$, $|E(H)| \geq 2d|V(H)| \geq 2d(|V(H)| - d - \frac{1}{2})$. \square

From Claim 6.2, we conclude that $|V(H)| \geq 2d + 1$ and $|E(H)| \geq 2d(|V(H)| - d - \frac{1}{2})$. 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 $|V(H)| \geq 2d + 1$ and $|E(H)| \geq 2d(|V(H)| - d - \frac{1}{2})$. 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 $|V(H)| = 2d + 1$, $|E(H)| \geq 2d(2d + 1 - d - \frac{1}{2}) = 2d(d + \frac{1}{2}) = \binom{2d+1}{2}$. 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 $\deg_H(v) \leq 2d$. Then $|V(H - v)| \geq 2d + 1$ and $|E(H - v)| \geq |E(H)| - 2d \geq 2d(|V(H - v)| - d - \frac{1}{2})$. 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 d . 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 $\deg_H(s), \deg_H(t) \geq 2d + 1$, we have that $|A| \geq 2d + 1$ and $|B| \geq 2d + 1$. 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 $H[A]$ and $H[B]$. Then, we have the following:

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

This is a contradiction to the fact that $|E(H)| \geq 2d(|V(H)| - d - \frac{1}{2})$. Therefore, either $H[A]$ or $H[B]$ satisfies 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. \square

The above proof can easily be turned into 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 $V(H)$. 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 finds 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 $O(mn^2 \log n)$ 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 $O(mn)$ and takes additional $O(n^2 m \log n)$ time to output a $(d + 1)$ -connected subgraph of H . Thus, the total running time of this algorithm is $O(n^2 m \log n)$.

We are now ready 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 smaller 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 $4d - 1 \geq 4(d - 1)$. 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, $|V(H)| \geq d + 1$. Since, $k \leq d$, 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^\star 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^\star 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 $G^\star[Y]$ is a clique. Hence, the degeneracy of G^\star is at least $4d - 1$. For the proof of the second, we have already argued that the size of $V(H)$ is at least $d + 1$ and that $V(H)$ is a k -connected set in G^\star . We will now prove $V(H)$ is a k -connected set in G . \square

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 smaller than k , there is a path from u to v in $G - C$. Since H is a k -connected subgraph of G^\star , there is a path from u to v in $G^\star - C$. Let $w_1 w_2 \dots w_\ell$, where $w_1 = u$ and $w_\ell = v$, be a path from w_1 to w_ℓ in $G^\star - C$. Since for any $i \in [\ell - 1]$, $w_i w_{i+1} \in E(G^\star)$, 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 . \square

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

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 $O(|T| \cdot n(n^{\frac{2}{3}} + m) + n^4 \log n)$ 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 computes 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 $|Y| \geq d + 1$ 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 $O(|T| \cdot n(n^{\frac{2}{3}} + m) + n^4 \log n)$. \square

LEMMA 6.11. *There is an algorithm that given as input a graph G , a set of terminal pairs T and $k \in \mathbb{N}$, runs in time $O(|T| \cdot n(n^{\frac{2}{3}} + m) + n^4 \log n)$ 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 $|S| \leq k$, 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, let $S \subseteq V(G)$ be such that $|S| \leq k$ 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 $S' \subsetneq S$ 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 $S \subseteq V(G) \setminus \{v\}$ be 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 be a minimal multicut of T in G , as otherwise 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 $T \setminus T'$ contain v . Let $S' \subseteq S \cup \{v\}$ be a minimal multicut of T in G . Note that $v \in S'$ and $|S'| \leq k + 1$. This contradicts the fact that v is an irrelevant vertex for $(G, T, k + 1)$. \square

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

Stable Multicut on General Graphs. Using our algorithm of Theorem 2 for STABLE MULTICUT on bounded degeneracy graphs and the Degeneracy Reduction Lemma (Lemma 6.5), we are now ready to 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 $k, d \in \mathbb{N}$ be such that $k \leq d + 1$. Let G be a graph that 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 $k \leq d + 1$ and $|V(H)| \geq d + 2$, we have that $V(H)$ is a k -connected set in G of size at least $d + 2$, which is a contradiction. \square

THEOREM 6. *STABLE MULTICUT can be solved in time $2^{O(k^3)} \cdot n^6 \log n$.*

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 $64^{k+2} \cdot 16(k + 2)^2 - 1$. Then, using Theorem 2, we get the solution. The running time of the algorithm follows from Lemma 6.5 and Theorem 2. \square

7 CONCLUSION

In this article, 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 $((\binom{d+1}{k}) \cdot k(d + 1))^{-1}$. 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 $T = \{\{s_1, t_1\}, \{s_2, t_2\}, \dots, \{s_\ell, t_\ell\}\}$ of terminal pairs, and an integer k returns an induced subgraph G^* 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 $2^{O(k)}$. Our new tools yield 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. [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 that is contained in the union of the r chosen sets. One of the most natural directions to pursue further is to find more applications of our tools than given in the article. Apart from this there are several natural questions that arise from 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, if \mathcal{S} is 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 independent sets, such that k - \mathcal{G} covering family of FPT size exists?
- (3) Observe that for all the problems whose non-stable version admits a $2^{O(k)}n^{O(1)}$ time algorithm on general graphs, such as s - t SEPARATOR and ODD CYCLE TRANSVERSAL, we get a $2^{O(k)}n^{O(1)}$ time algorithm for these problems on graphs of bounded degeneracy. As a corollary, we get a $2^{O(k)}n^{O(1)}$ 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 a $2^{O(k)}n^{O(1)}$ time algorithm on general graphs.

ACKNOWLEDGMENT

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

REFERENCES

- Béla Bollobás and Andrew Thomason. 1998. Proof of a conjecture of Mader, Erdős, 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 *Proceedings of the Symposium on the Theory of Computing (STOC'11)*. 459–468.
- Andreas Brandstadt, Jeremy P. Spinrad, et al. 1999. *Graph Classes: A Survey*. Vol. 3. Siam.
- Nader H. Bshouty and Ariel Gabizon. 2017. Almost optimal cover-free families. In *Proceedings of the International Conference on Algorithms and Complexity*. Springer, 140–151.
- 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. *J. ACM* 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 Trans. Algor.* 11, 4 (2015), 28.
- Rajesh Hemant Chitnis, Mohammad Taghi 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.
- Hyeong-Ah Choi, Kazuo Nakajima, and Chong S. Rim. 1989. Graph bipartization and via minimization. *SIAM J. Discrete Math.* 2, 1 (1989), 38–47.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. The MIT Press.

- Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer.
- Marek Cygan, Łukasz Kowalik, and Marcin Pilipczuk. 2013a. Open problems from the update meeting on graph separation problems. <http://worker2013.mimuw.edu.pl/slides/update-opl.pdf>.
- Marek Cygan, Marcin Pilipczuk, Michał 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 J. Comput.* 23 (1994), 864–894.
- Erik D. Demaine, Gregory Gutin, Dániel Marx, and Ulrike Stege. 2007. 07281 open problems—Structure theory and FPT algorithms for graphs, digraphs and hypergraphs. In *Structure Theory and FPT Algorithms for Graphs, Digraphs and Hypergraphs*, 08.07.2007–13.07.2007.
- R. Diestel. 2000. *Graph Theory* (2nd ed.). Springer, Berlin.
- Lester R. Ford and Delbert R. Fulkerson. 1956. Maximal flow through a network. *Canad. J. Math.* 8, 3 (1956), 399–404.
- Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. 2013. Characterisations of nowhere dense graphs (invited talk). In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13) (LIPIcs)*, Vol. 24. 21–40.
- Jianxiu Hao and James B. Orlin. 1992. A faster algorithm for finding the minimum cut in a graph. In *Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 165–174.
- Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. 2014. Linear-time FPT algorithms via network flow. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. 1749–1761.
- Richard M. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Springer, 85–103.
- Ken-ichi Kawarabayashi and Bruce A. Reed. 2010. An (almost) linear time algorithm for odd cycles transversal. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*. 365–378.
- János Komlós and Endre Szemerédi. 1996. Topological cliques in graphs II. *Combin. Prob. Comput.* 5 (1996), 79–90.
- Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Magnus Wahlström. 2015. Fixed-parameter tractability of multicut in directed acyclic graphs. *SIAM J. Disc. Math.* 29, 1 (2015), 122–144.
- Stefan Kratsch and Magnus Wahlström. 2012. Representative sets and irrelevant vertices: New tools for kernelization. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS'12)*. 450–459.
- Daniel Lokshtanov and Dániel Marx. 2013. Clustering with local restrictions. *Inf. Comput.* 222 (2013), 278–292.
- Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. 2014. Faster parameterized algorithms using linear programming. *ACM Trans. Algor.* 11, 2 (2014), 15:1–15:31.
- Daniel Lokshtanov and M. S. Ramanujan. 2012. Parameterized tractability of multiway cut with parity constraints. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'12)*. 750–761.
- Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. 2015. Linear time parameterized algorithms for subset feedback vertex set. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'15)*. 935–946.
- Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. 2016a. A linear time parameterized algorithm for directed feedback vertex set. *CoRR abs/1609.04347* (2016).
- Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. 2016b. A linear time parameterized algorithm for node unique label cover. *CoRR abs/1604.08764* (2016).
- W. Mader. 1972. Existenz n -fach zusammenhängender Teilgraphen in Graphen genügend großer Kantendichte. *Abhand. Math. Sem. Univ. Hamburg* 37, 1 (1972), 86–97.
- Dániel Marx. 2006. Parameterized graph separation problems. *Theor. Comput. Sci.* 351, 3 (2006), 394–406.
- Dániel Marx. 2011. Important separators and parameterized algorithms. In *Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 5–10.
- Dániel Marx, Barry O'Sullivan, and Igor Razgon. 2013. Finding small separators in linear time via treewidth reduction. *ACM Trans. Algor.* 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. *J. ACM* 30, 3 (1983), 417–427.
- Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. 2012. On parameterized independent feedback vertex set. *Theor. Comput. Sci.* 461 (2012), 65–75.
- Jaroslav Nešetřil and Patrice Ossana de Mendez. 2008. Grad and classes with bounded expansion I. Decompositions. *Eur. J. Combin.* 29, 3 (2008), 760–776.
- Jaroslav Nešetřil and Patrice Ossana de Mendez. 2009. From sparse graphs to nowhere dense structures: Decompositions, independence, dualities, and limits. In *Proceedings of the European Congress of Mathematics*. 135–165.

- Jaroslav Nešetřil and Patrice Ossona de Mendez. 2011. On nowhere dense graphs. *Eur. J. Combin.* 32, 4 (2011), 600–617.
- Jaroslav Nešetřil and Patrice Ossona de Mendez. 2012. *Sparsity—Graphs, Structures, and Algorithms*. (Algorithms and Combinatorics, Vol. 28). Springer.
- Marcin Pilipczuk and Magnus Wahlström. 2018. Directed multicut is $W[1]$ -hard, even for four terminal pairs. *ACM Trans. Comput. Theor.* 10, 3 (2018), 13.
- M. S. Ramanujan and Saket Saurabh. 2014. Linear time parameterized algorithms via skew-symmetric multicuts. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. 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. *J. ACM* 44, 4 (1997), 585–591.
- Benny Sudakov. 2016. Graph theory. *Lect. Notes* (2016). http://www2.math.ethz.ch/education/bachelor/lectures/fs2016/math/graph_theory/graph_theory_notes.pdf.

Received July 2018; revised January 2020; accepted January 2020