
Proceedings Paper:
Aute, S., Panolan, F., Saha, S. et al. (2025) Parameterized Complexity of Generalizations of Edge Dominating Set. In: 50th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2025). 50th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2025), 20-23 Jan 2025, Bratislava, Slovakia. Lecture Notes in Computer Science, 15538. Springer, Cham, Switzerland, pp. 65-79. ISBN: 978-3-031-82669-6. ISSN: 0302-9743. EISSN: 1611-3349.

https://doi.org/10.1007/978-3-031-82670-2_6

Parameterized Complexity of Generalizations of Edge Dominating Set

Shubhada Aute[1], Fahad Panolan[2], Souvik Saha[3], Saket Saurabh[3,4], and Anannya Upasana[3]

[1] Department of Computer Science and Engineering, IIT Hyderabad, India.
cs21resch11001@iith.ac.in
[2] School of Computer Science, University of Leeds, UK
f.panolan@leeds.ac.uk
[3] The Institute of Mathematical Sciences, HBNI, Chennai, India
{souviks, saket, anannyaupas}@imsc.res.in
[4] University of Bergen, Norway

Abstract. The objective of this article is to propose two natural generalizations of covering edges by edges (Edge Dominating Set) and study these problems from the multivariate lens. The first is simply considering Edge Dominating Set on hypergraphs, called Hyperedge Dominating Set. Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{F})$, a set $F \subseteq \mathcal{F}$ is called a *hyperedge dominating set* if all hyperedges intersect with at least one hyperedge $e \in F$. The objective of the Hyperedge Dominating Set problem is to determine whether a hyperedge dominating set of size at most k exists. We find it quite surprising that such generalization is missing from the literature. The second extension we consider is the t-Path Edge Dominating Set problem. In this problem, the input consists of a graph G and an integer k, and the goal is to find a set $\mathcal{P}$ of at most k paths, each of length at most t, such that for every edge in G, at least one of its endpoints belongs to the vertex set $V(P)$ for some $P \in \mathcal{P}$. We show the following results and add to the literature on Edge Dominating Set.

- Hyperedge Dominating Set is FPT parameterized by $k+d$, where d is the maximum size of a hyperedge in the input hypergraph.
- A kernel of size $\mathcal{O}(k^d)$ can be obtained for the Hyperedge Dominating Set problem, where d is the maximum size of a hyperedge in the input hypergraph.
- The problem of finding a Hyperedge Dominating Set is computationally difficult; specifically it is W[2]-hard when parameterized by k. This hardness result holds even when each vertex is contained in at most 2 hyperedges and the intersection between any two hyperedges is at most 1.
- t-Path Edge Dominating Set is FPT when parameterized by $k+t$. Additionally, it has a kernel of size $\mathcal{O}(k^3t^3)$.

Keywords: Edge Dominating Set · FPT · Hypergraph · Kernel.

1 Introduction

Covering things by things is ubiquitous in theoretical computer science. In most cases, these can be abstracted as either the classical Set Cover problem or the Hitting Set problem. In these problems, we are given a hypergraph $(\mathcal{U}, \mathcal{F})$, here $\mathcal{U} = \{u_1, \ldots, u_n\}$ is a universe (set of vertices), and $\mathcal{F} = \{F_1, \ldots, F_m\}$ is a family of subsets over $\mathcal{U}$ also called hyperedges, and a positive integer k. In the Set Cover problem, the goal is to find a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size at most k, the union of which contains all the elements of $\mathcal{U}$. In the Hitting Set problem, we want to find a subset $\mathcal{U}' \subseteq \mathcal{U}$ of size at most k that has a non-empty intersection with each element of $\mathcal{F}$. These problems, together with their numerous variants and generalizations, have been some of the most explored research directions in the field of parameterized complexity. Motivated by these studies, in this paper, we consider two generalizations of classical Edge Dominating Set and study them from the Parameterized Complexity perspective [3,9,14,19,20,22].

In the Edge Dominating Set (EDS) problem, we are given a graph G and an integer k, and the task is to find a set of k edges that dominate all the edges in G. We say that an edge $e_1 = (a_1, b_1)$ dominates (or covers) another edge $e_2 = (a_2, b_2)$ if $\{a_1, b_1\} \cap \{a_2, b_2\} \neq \emptyset$. Note that the edge dominating set is a dominating set in the line graph. This problem is known to be NP-Complete even when restricted to planar or bipartite graphs with maximum degree 3 [32]. It is also known to admit a factor 2 approximation algorithm in polynomial time [13]. Fernau was the first to consider the problem from the perspective of parameterized complexity and, by employing enumeration-based techniques, obtained an FPT algorithm [10]. After a series of improvements, the current best algorithm was given by Iwaide and Nagamochi which runs in time $\mathcal{O}^\star(2.2351^k)$ [18][5]. The problem also admits a polynomial kernel with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^3)$ edges [30]. In fact, people have also tried to improve constants, and the current best-known kernel has $\max\{\frac{k^2}{2} + \frac{7k}{2}, 6k\}$ vertices and $\frac{8k^3}{27} + \mathcal{O}(k^2)$ edges [15].

Escoffier et al. even considered this problem in the world of FPT approximation and designed an FPT algorithm that is faster than the best known FPT exact algorithm and has a ratio better than 2. In fact, they give an FPT-approximation scheme [8]. Finally, we would also like to mention that several exact algorithms have been made for the problem [11,27,28,31]. This demonstrates the extensive research on Edge Dominating Set through algorithmic approaches designed to address its NP-hardness.

1.1 Edge Dominating Set on Hypergraphs

We first define the notion of domination by a vertex or an edge in a graph. A *vertex dominates* itself, all its neighbors, and all the edges that are incident with it. Similarly, an *edge dominates* its two endpoints, and all the edges incident with either of its endpoints. These problems can be broadly classified into four

[5] The $\mathcal{O}^\star$ notation hides polynomial factors.

Dominator \ Dominatee	Vertex	Edge / Hyperedge
Vertex	Dominating Set	Vertex Cover Hitting Set
Edge / Hyperedge	Edge Cover Set Cover	Edge Dominating Set Hyperedge Dominating Set*

Fig. 1. An overview of domination problems in graphs and their counterparts in hypergraphs. Problem marked with $*$ has not been previously studied.

Dominatee	Dominator	Problem	FPT Status
In Graphs			
Vertices	Vertices	Dominating Set	W[2]-hard [6]
Vertices	Edges	Edge Cover	P [25]
Edges	Edges	Edge Dominating Set	FPT [10,18,30]
Edges	Vertices	Vertex Cover	FPT [4,17]
In Hypergraphs (d-Hypergraphs)			
Vertices	Vertices	Dominating Set	W[2]-hard [6]
Vertices	Hyperedges	Set Cover (d-Set Cover)	W[2]-hard [5] (FPT [5])
Hyperedges	Vertices	Hitting Set (d-Hitting Set)	W[2]-hard [7] (FPT [24,2])
Hyperedges	Hyperedges	Hyperedge Dominating Set (d-Hyperedge Dominating Set)	W[2]-hard (FPT)

Table 1. The figure shows the FPT status with respect to the parameter k (solution size) of the various domination problems in graphs and hypergraphs.

categories based on whether the dominator and the dominatee is a vertex set or an edge set. Each type gives rise to some classical well-studied problems in graph algorithms. A brief overview of the classification is given in fig. 1. In particular, a set of vertices dominating all the vertices is the Dominating Set problem, a set of vertices dominating all the edges is the Vertex Cover problem, a set of edges dominating all the vertices is the Edge Cover problem, and a set of edges dominating all the edges is the Edge Dominating Set problem. Our first generalization is obtained by considering these basic problems from graphs to hypergraphs.

By considering a natural generalization of these problems on hypergraphs (d-hypergraphs), we get another set of well-studied problems. A d-hypergraph is a hypergraph where each hyperedge has size at most d, called a d-hyperedge. In the case where we want to dominate all the edges (d-hyperedges) by a set of vertices, we get the Hitting Set (d-Hitting Set) problem. Also, when all the vertices need to be covered by a set of hyperedges (d-hyperedges), we get the Set Cover (d-Set Cover) problem. The status of each of these problems with

respect to parameterized complexity is mentioned in Table 1. The domination of vertices by vertices remains an equivalent problem in terms of complexity in hypergraphs. Since Dominating Set in graphs is W[2]-hard parameterized by k, Vertex Dominating Set in hypergraphs is also W[2]-hard parameterized by k. However, the case where all the hyperedges need to be dominated by a set of hyperedges has not been studied. We fill this gap in our knowledge by studying the Hyperedge Dominating Set (HEDS) problem through the lens of parameterized complexity. In particular, we prove the following.

Theorem 1 ($\star$[6]). *There is an algorithm of running time $\mathcal{O}(d^{dk} \cdot 2^{dk}|\mathcal{F}|)$ for d-Hyperedge Dominating Set.*

Our algorithm is inspired by the techniques for Edge Dominating Set problem studied in these papers [10,18,30].

Theorem 2. *d-Hyperedge Dominating Set admits a kernel of size $\mathcal{O}((dk)^{d^2})$.*

Theorem 2 is proved using the concept of representative sets [12].Given a hypergraph H defined on $\mathcal{U} = \{u_1, \ldots, u_n\}$ and $\mathcal{F} = \{F_1, \ldots, F_m\}$, we can define a dual hypergraph H^* on $\mathcal{U}^*$ and $\mathcal{F}^*$ as follows. For each $F \in \mathcal{F}$, create a vertex $x_F \in \mathcal{U}^*$ and for each $u \in \mathcal{U}$, create a subset of $\mathcal{U}^*$ corresponding to the hyperedges in $\mathcal{F}$ that contain u and add this subset to $\mathcal{F}^*$. Notice that a set of vertices S of a hypergraph H is a vertex dominating set in H if and only if the set of hyperedges corresponding to S is a hyperedge dominating set in the dual hypergraph H^*. Specifically, $S \subseteq \mathcal{U}$ is a vertex dominating set of size at most k in a hypergraph H where the degree of every vertex is bounded by d if and only if the set of at most k d-hyperedges corresponding to S is a hyperedge dominating set in the dual d-hypergraph H^*. Thus, our FPT algorithm and kernelization results for d-Hyperedge Dominating Set (d-HEDS) also extend to Vertex Dominating Set in hypergraphs where the degree of every vertex is bounded by d. Notice that even though the degree of every vertex in H^* is at most d, the size of the hyperedges in H^* is not bounded. So, a simple branching algorithm may not work.

However, when we allow the size of the hyperedges to be unbounded, the problem becomes W[2]-hard. In fact, we show that the problem Hyperedge Dominating Set is hard even for a specific case where the frequency of every element is bounded, i.e., every vertex has bounded degree and any pair of sets in the family has a bounded intersection.

Theorem 3 ($\star$). Hyperedge Dominating Set *is* W[2]-hard *parameterized by k even when the intersection of any two edges is bounded by* 1 *and the degree of any vertex is bounded by* 2.

We show that the above theorem implies that Hyperedge Dominating Set is W[2]-hard even when the hypergraph is $K_{2,2}$-free. Here $K_{2,2}$-free means that there are no two vertices present in two hyperedges. This result is in contrast to the fact that Dominating Set is FPT in $K_{i,j}$-free graphs, that is, graphs that do not contain the complete bipartite graph with i and j vertices as subgraph [26].

[6] Proofs of results marked with $\star$ are omitted due to paucity of space.

1.2 Covering Edges by Paths : t-path Edge Dominating Set

Our next generalization is motivated by the following analogy: Vertex Cover and Edge Dominating Set can be viewed as *domination* of edges by paths of length 1 and 2, respectively. Here, we consider a vertex as a path of length 1 and an edge as a path of length 2. We extend this to study *domination* of edges by paths of length t. In this problem, given a graph G and an integer k as input, the goal is to find at most k paths, each with length at most t, which dominate all edges in the graph. In simple words, deleting vertices appearing on these paths results in an independent set. In this paper, we design an FPT algorithm and a kernel for the t-path Edge Dominating Set (t-path EDS) problem. We give a randomized FPT algorithm parameterized by $k+t$ using the technique of color coding, and then we derandomize it by utilizing an (n,k)-perfect hash family [5].

Theorem 4. *Given an instance $\mathcal{I} = (G = (V, E), k)$ of t-path Edge Dominating Set, there is a deterministic FPT algorithm that runs in time $(8e)^{kt}4^t 2^{\mathcal{O}(log^2 kt)} n^{\mathcal{O}(1)}$ and finds at most k paths, each of length at most t, that dominate all the edges in E.*

One might ponder why we do not study this problem parameterized by k or t alone? We give a reduction from an NP-hard problem s-t Hamiltonian Path to our problem. In the s-t Hamiltonian Path problem, we are given a graph G, two vertices $s, t \in V(G)$ and the task is to check if a path exists that goes through every vertex exactly once and starts from s and ends at t. The reduction is as follows. Given G, we subdivide every edge. We add paths $P_1 = (s_1, s_2, s)$ and $P_2 = (t, t_1, t_2)$ to the graph with subdivided edges. Let the modified graph be G'. Then, $(G', 1, 2n + 1)$ is an equivalent instance of t-path Edge Dominating Set. It is easy to see that G has a s-t hamiltonian path if and only if G' has a path on $2n+1$ vertices that dominates all its edges. An FPT algorithm for t-path Edge Dominating Set parameterized by k alone would imply a polynomial time algorithm for the s-t Hamiltonian Path, implying P = NP.

Similarly, an FPT algorithm parameterized by t alone would imply a polynomial time algorithm for Vertex Cover, which is our problem corresponding to $t = 1$ and is known to be NP-hard, thus, implying P = NP.

Theorem 5 ($\star$). *t-path Edge Dominating Set admits a kernel of size $\mathcal{O}(k^3 t^3)$.*

We would like to remark that the best known kernel for Vertex Cover (corresponding to $t = 1$) has size $\mathcal{O}(k)$ and for Edge Dominating Set (corresponding to $t = 2$) has size $\mathcal{O}(k^2)$ [15,21,29].

2 Preliminaries

We use $\mathbb{N}$ to denote the set of positive integers. For a graph G, we denote its vertex set and edge set as $V(G)$ and $E(G)$, respectively. A hypergraph $\mathcal{H}$ on

a universe $\mathcal{U} = \{u_1, \ldots, u_n\}$ is a family $\mathcal{F}$ of subsets of $\mathcal{U}$. For a set $W \subseteq \mathcal{U}$, $\mathcal{F} - W$ denotes all the sets in $\mathcal{F}$ that do not contain any element from W. The frequency or equivalently the degree of a vertex v in a hypergraph is the number of hyperedges which contains v. A path P in a graph G is a sequence of distinct vertices $v_1, \ldots, v_\ell$, with $\ell > 1$, such that $(v_i, v_{i+1}) \in E(G)$, for each $i \in [\ell - 1]$. We say that a path is of length t if the path contains t distinct vertices. For a path P, let $V_P = \{v : v \in V(P)\}$ be the set of vertices contained in the path P. For a set of paths $\mathcal{P} = \{P_1, \ldots, P_\ell\}$, let $V_{\mathcal{P}} = \{v : v \in \bigcup_{P \in \mathcal{P}} V(P)\}$ be the set of vertices contained in the paths in $\mathcal{P}$. Let $\mathcal{P}$ be a set of paths that dominate all the edges in a graph, then we denote the set of vertices in $V_{\mathcal{P}}$ as *used*. For a vertex v, $N(v) = \{u : (u, v) \in E\}$ denotes the set of neighbors of v. For a set of vertices X, $N(X) = (\bigcup_{v \in X} N(v)) \setminus X$. For a subset of vertices S, $G[S]$ denotes the graph induced on S, i.e., the graph on the vertex set S and the set of edges present between any two vertices of S.

For two sets A and B, $A \setminus B$ denotes the set of elements in A, but not in B. For an integer i, we denote the set $\{1, \ldots, i\}$ by $[i]$. For integers i and j, we denote the set $\{i, \ldots, j\}$ by $[i, j]$. For sets A and B, by $A \uplus B$, we denote the disjoint union of the sets.

3 Kernel for d-Hyperedge Dominating Set

In this section, we will develop a kernel for d-Hyperedge Dominating Set using representative sets. We begin with the definition of representative sets.

Definition 1 ([12]). *(q-**Representative Family**) Given a family $\mathcal{A}$ over a universe $\mathcal{U}$, a subfamily $\mathcal{A}' \subseteq \mathcal{A}$ is said to q-represent $\mathcal{A}$ if for every set $B \subseteq \mathcal{U}$ of size q such that there is an $A \in \mathcal{A}$ and $A \cap B = \emptyset$, there is a set $A' \in \mathcal{A}'$ such that $A' \cap B = \emptyset$. If $\mathcal{A}' \subseteq \mathcal{A}$ is q-representative for $\mathcal{A}$ we write $\mathcal{A}' \subseteq^q_{rep} \mathcal{A}$.*

Theorem 6 ([12]). *Given a family $\mathcal{A}$ of sets of size p over a universe, and $q \in \mathbb{N}$, a q-representative family $\hat{\mathcal{A}} \subseteq \mathcal{A}$ for $\mathcal{A}$ with at most $\binom{p+q}{p}$ sets can be computed in time $\mathcal{O}(|\mathcal{A}|(\binom{p+q}{p}p^\omega + \binom{p+q}{p}^{\omega-1})$. Here, $\omega < 2.373$ is the matrix multiplication exponent.*

Theorem 2. *d-Hyperedge Dominating Set admits a kernel of size $\mathcal{O}((dk)^{d^2})$.*

Proof. In the d-HEDS problem, we have a universe $\mathcal{U}$ and family $\mathcal{F}$ of sets of cardinality at most d and an integer k as input. We compute a q-representative set, with $q = dk$, of $\mathcal{F}$ using the algorithm in Theorem 6. Let $\hat{\mathcal{F}}$ be the q-representative set of $\mathcal{F}$. We know $|\hat{\mathcal{F}}| \leq \binom{dk+d}{d} = \mathcal{O}((dk)^d)$ and we can find $\hat{\mathcal{F}}$ in time $\mathcal{O}(|\mathcal{F}|((dk)^d d^\omega + (dk)^{d\omega - d})) = \mathcal{O}(|\mathcal{F}|(dk)^{d\omega})$. Note that $|\mathcal{F}| \geq \binom{dk+d}{d}$, otherwise, we can take $\hat{\mathcal{F}} = \mathcal{F}$. Now, let $\hat{\mathcal{U}}$ be the union of elements in the sets in $\hat{\mathcal{F}}$. Then, $|\hat{\mathcal{U}}| \leq d \cdot \binom{dk+d}{d}$. For all possible subsets U of $\hat{\mathcal{U}}$, of size at most d, we add a set from $\mathcal{F}$ to $\hat{\mathcal{F}}$, if it exists, which contains the elements of U. The number of sets added to $\hat{\mathcal{F}}$ is at most $\binom{|\hat{\mathcal{U}}|}{d} + \binom{|\hat{\mathcal{U}}|}{d-1} + \ldots + \binom{|\hat{\mathcal{U}}|}{1} \leq \binom{|\hat{\mathcal{U}}|+d-1}{d} \leq \binom{d\binom{dk+d}{d}+d-1}{d}$.

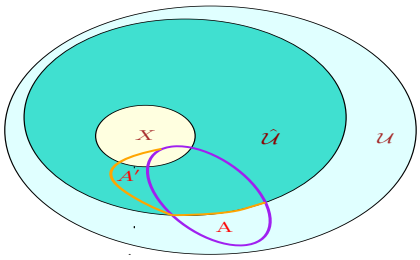

Fig. 2. An illustration of the proof of Theorem 2.

Note that the size of $\hat{\mathcal{F}}$ now is at most $\binom{dk+d}{d} + \binom{d\binom{dk+d}{d}+d-1}{d} = \mathcal{O}((dk)^{d^2})$. Note that we also assume $|\mathcal{F}| \geq \mathcal{O}((dk)^{d^2})$, otherwise we can return the original instance itself. We claim that $(\hat{\mathcal{U}}, \hat{\mathcal{F}})$ is our desired kernel. The total time taken is $\mathcal{O}(|\mathcal{F}|((dk)^{d^2}))$. Our construction is polynomial in the size of the input instance.

Let $\hat{S}$ be a solution for the instance $(\hat{\mathcal{U}}, \hat{\mathcal{F}})$ and $\mathcal{U}_{\hat{S}} = \bigcup_{F \in \hat{S}} F$. Then, $\mathcal{U}_{\hat{S}}$ hits all the sets in $\hat{\mathcal{F}}$. We claim that $\mathcal{U}_{\hat{S}}$ hits all the sets in $\mathcal{F}$. Suppose not. Then let A be a set in $\mathcal{F}$ that is not hit by $\mathcal{U}_{\hat{S}}$. Since $\hat{\mathcal{F}}$ contains a q-representative set, there must exist some set A' in $\hat{\mathcal{F}}$ which is disjoint from $\mathcal{U}_{\hat{S}}$. But $\mathcal{U}_{\hat{S}}$ hits all the sets in $\hat{\mathcal{F}}$, which is a contradiction.

Let S be a solution for the instance $(\mathcal{U}, \mathcal{F})$ and $\mathcal{U}_S = \bigcup_{F \in S} F$. We know that S dominates all the sets in $\mathcal{F}$. We claim that there exists a subfamily S' in $\hat{\mathcal{F}}$ of cardinality at most $|S|$ that dominates all the sets in $\hat{\mathcal{F}}$. For a set $A \in S$, if $A \in \hat{\mathcal{F}}$ then we pick A in S'. Otherwise, we have two cases. If $\hat{\mathcal{U}} \cap A \neq \emptyset$, then we choose a set A' that is added for the subset $\hat{\mathcal{U}} \cap A$ in the construction of $\hat{F}$. If $\hat{\mathcal{U}} \cap A = \emptyset$, we don't add any set corresponding to A. This completes the construction of S' and notice that $|S'| \leq |S|$. Next we need to prove that S' is a solution to $(\hat{\mathcal{U}}, \hat{\mathcal{F}})$. Fix an arbitrary set $X \in \hat{\mathcal{F}}$. We know that X is dominated by S. Let $A \in S$ be a set that dominates X. That is, $X \cap A \neq \emptyset$. If $A \in \hat{\mathcal{F}}$, then $A \in S'$ (by construction of S'), and hence X is dominated by S'. Suppose this is not the case. Notice that $\emptyset \neq X \cap A \subseteq A \cap \hat{\mathcal{U}}$. We have added a set A' to S' corresponding to A, where $A' \cap \hat{\mathcal{U}} = A \cap \hat{\mathcal{U}}$. This implies that A' dominates X. See fig. 2 for an illustration. □

4 FPT algorithm for t-path Edge Dominating Set

In this section, we present an algorithm for solving t-path Edge Dominating Set (t-path EDS). Given a graph $G = (V, E)$ and an integer k, the objective of this problem is to find at most k paths, each with a length of at most t, that cover all the edges in G. Recall that a path is considered to have a length of t if it consists of t vertices. A path P is said to cover (dominate) an edge $e = (a, b)$ if the set of vertices V_P intersects with $\{a, b\}$. It is important to note that when

$t = 2$, the problem is the same as the Edge Dominating Set problem. The paths in the solution may have overlapping vertices or edges.

A randomized algorithm for t-path EDS is described in Algorithm 3 and Theorem 7 states the main result. However, before we state our main result, we list a few lemmas that will be used to prove the correctness of our algorithm.

Lemma 1 ([16]). *There is an algorithm to find a vertex cover of size k (if it exists) in a graph G, in time $\mathcal{O}^\star(1.25284^k)$.*

Our next lemma uses the notion of a *colorful path*. We first define what we mean by a colorful path in a graph and mention a known result that states how to find one, if it exists.

Definition 2 (Colorful Path). *For a graph $G = (V, E)$, a positive integer k and a coloring $f : V \to [k]$ of the vertices of G using k colors, we call P a colorful path in G if all the vertices in P get distinct colors. That is, for any two distinct vertices $u, v \in V_P$, $f(u) \neq f(v)$.*

Lemma 2 ([1,5]). *Let G be a graph, k be a positive integer, and $f : V \to [k]$ be a coloring of V using k colors. There exists a deterministic algorithm that checks in time $2^k n^{\mathcal{O}(1)}$ whether G contains a colorful path on k vertices and, if this is the case, returns one such path.*

For a graph G, a set of colors $\mathcal{C}ol$, and a coloring $f : V \to [|\mathcal{C}ol|]$, we define a subroutine called $\mathsf{Path}(\mathcal{C}ol)$ which is TRUE if invoking Lemma 2 on the instance $(G, |\mathcal{C}ol|, f)$ returns a colorful path that uses all the colors in $\mathcal{C}ol$. For implementation details, please refer to [5].

4.1 colorful t-path cover and t-path cover

We describe a problem called colorful t-path cover that takes a graph G, a positive integer k, a vertex subset $V' \subseteq V$, and a coloring $f : V' \to [q]$ of V' using q colors, and checks if there exist at most k colorful paths in $G[V']$, each of length at most t, say $\mathcal{P}$, such that $\{f(v) : v \in V_{\mathcal{P}}\} = [q]$. The paths need not be vertex disjoint which implies that $q \leq kt$. In essence, we want to find at most k colorful paths, each of length at most t, that use all the colors. We define it formally below.

colorful t-path cover
Input: $\mathcal{I} = (G, k, V', q, f)$ where $G = (V, E)$ is a graph, $k \in \mathbb{N}$, $V' \subseteq V$, a set of q colors, and $f : V' \to [q]$ is a coloring of V' using q colors.
Parameter: $k + t$.
Output: A set $\mathcal{P}$ such that $|\mathcal{P}| \leq k$, each $P \in \mathcal{P}$ is a colorful path of length at most t and $\{f(v) : v \in V_{\mathcal{P}}\} = [q]$.

We can solve an instance $\mathcal{I} = (G, k, V', q, f)$ of colorful t-path cover by invoking Algorithm 1 that uses Lemma 3 as a subroutine.

Algorithm 1 An FPT algorithm for COLORFUL t-PATH COVER

Input: $(\mathcal{I} = (G = (V, E), k, V', \mathcal{C}ol, f))$
Parameter: $k + t$
Output: YES if there exist at most k paths in $G[V']$, each of length at most t, that together use all the colors in $\mathcal{C}ol$, else NO.

1: **for** $i = 1$ to k **do**
2: Invoke Lemma 3 on $(\mathcal{C}ol, i)$.
3: If the output is YES, then **return** YES.
4: **end for**
5: **return** NO.

Lemma 3 ($\star$). *Let $G = (V, E)$ be a graph, $V' \subseteq V$, $\mathcal{C}ol$ be a set of colors, k be a positive integer, and $f : V' \to [|\mathcal{C}ol|]$ be a function. Then, for a subset of colors $col \subseteq \mathcal{C}ol$ and some $i \in \mathbb{Z}^+$, $\mathsf{Cover}(col, i)$ is TRUE if there exist i paths, each of them colorful and of length at most t, in $G[V']$, say $\mathcal{P}$, such that $\{f(v) : v \in V_{\mathcal{P}}\} = col$. It is FALSE otherwise. $\mathsf{Cover}(col, i)$ can be computed using the following recursive formula.*

$$\mathsf{Cover}(col, i) = \bigvee_{\substack{C \subseteq col \text{ where } |C| \leq t, \\ col' = (col \setminus C) \cup W \text{ where } W \subset C, \text{ and} \\ \mathsf{Path}(C) = \mathsf{TRUE}}} \mathsf{Cover}(col', i-1)$$

with

$$\mathsf{Cover}(col, 1) = \begin{cases} \mathsf{Path}(col) & \text{if } |col| \leq t \\ \mathsf{FALSE} & \text{otherwise} \end{cases}$$

Moreover, $\mathsf{Cover}(\mathcal{C}ol, k)$ can be computed in time $2^{2(kt+t)} \cdot k^2 t \cdot n^{\mathcal{O}(1)}$.

The running time of Algorithm 1 is $2^{2(kt+t)} \cdot \mathcal{O}(k^3 t) \cdot n^{\mathcal{O}(1)}$, since Lemma 3 is invoked at most k times. Its correctness follows from the correctness of the above lemma.

We now describe another problem called t-PATH COVER that takes a graph G, an integer k, a partition $V_1 \uplus V_2 \uplus V_3$ of V, and checks if there exist at most k paths in $G[V_1 \cup V_2]$, each of length at most t, that cover all the vertices in V_1 but do not use any vertex from V_3. The paths need not be vertex disjoint. We define it formally below.

t-PATH COVER
Input: An instance $\mathcal{I} = (G, k, V_1, V_2, V_3)$ where $G = (V, E)$ is a graph, k is a positive integer, and $V_1 \uplus V_2 \uplus V_3$ is a partition of V.
Parameter: $k + t$
Output: A set $\mathcal{P}$ such that $|\mathcal{P}| \leq k$, each $P \in \mathcal{P}$ is a path of length at most t in $G[V_1 \cup V_2]$, $V_1 \subseteq V_{\mathcal{P}}$ and $V_3 \cap V_{\mathcal{P}} = \emptyset$.

Algorithm 2 An FPT algorithm for t-PATH COVER

Input: $(\mathcal{I} = (G = (V, E), k, V_1, V_2, V_3))$
Parameter: $k + t$
Output: YES if there exist at most k paths in $G[V_1 \cup V_2]$, each of length at most t, that cover all the vertices in V_1 and do not contain any vertex from V_3, else NO.

1: Color the vertices of V_1 such that every vertex gets a distinct color. Let $\widetilde{\chi} : V_1 \to [kt - |V_1| + 1, kt]$ be a coloring where vertices of V_1 get distinct colors.
2: **for** $r = 0$ to $kt - |V_1|$ **do**
3: Color the vertices in V_2 uniformly at random using r colors. Let $\chi : V_2 \to [r]$ be a coloring of V_2 uniformly at random. (Note that when $r = 0$, we do not color the vertices of V_2 and proceed with just the coloring on V_1.)
4: Let $\mathcal{C}ol = \mathsf{Range}(\chi) \cup \mathsf{Range}(\widetilde{\chi})$.
5: Define a coloring function $f : (V_1 \cup V_2) \to [|\mathcal{C}ol|]$ where $f(v) = \widetilde{\chi}(v)$ if $v \in V_1$ and $f(v) = \chi(v)$ if $v \in V_2$.
6: Invoke Algorithm 1 on the instance $(G, k, V_1 \cup V_2, \mathcal{C}ol, f)$.
7: If the algorithm outputs YES, then **return** YES.
8: **end for**
9: **return** NO instance.

Lemma 4 ($\star$). *Given an instance $\mathcal{I} = (G, k, V_1, V_2, V_3)$ of t-PATH COVER, there is a randomized algorithm that, given a* YES *instance, returns* YES *with probability at least $\left(1 - \frac{1}{e^c}\right)$ for some constant $c > 1$, and given a* NO *instance, always returns* NO*. The algorithm runs in time $2^{2(kt+t)} \cdot e^{kt} \cdot k^3 t^2 \cdot n^{\mathcal{O}(1)}$.*

A description of the algorithm is given in Algorithm 2.

4.2 Algorithm for t-PATH EDS

Our randomized FPT algorithm is described in Algorithm 3.

Theorem 7. *Given an instance $\mathcal{I} = (G = (V, E), k)$ of t-PATH EDS, there is a randomized* FPT *algorithm that runs in time $(8e)^{kt} 4^t k^4 t^2 n^{O(1)}$ and finds at most k paths, each of length at most t, that dominate all edges in E. Given a* YES *instance, it returns a solution with probability at least $\left(1 - \frac{1}{e^c}\right)$, for a constant $c > 1$. It returns* NO *when given a* NO *instance.*

Proof. We first compute a vertex cover A of size at most kt using Lemma 1. Recall that we call a vertex *used* if it occurs in one of the k solution paths. We guess a subset $A_{\text{used}} \subseteq A$ that consists of used vertices. Note that $|A_{\text{used}}|$ should be at most kt as the total number of used vertices cannot exceed kt. Then, $A_{\text{left}} = A \setminus A_{\text{used}}$ is the remaining part of the vertex cover A, and $I = V \setminus A$ is an independent set. Since vertices in A_{left} are not used, the vertices in $N_{\text{left}} = N(A_{\text{left}}) \cap I$ must be used to dominate the edges going across A_{left} and N_{left}. In case of a YES instance, $|N_{\text{left}}| \leq kt$. Moreover, for a YES instance, $G[A_{\text{left}}]$ must be an independent set because any edge completely contained in $G[A_{\text{left}}]$ can only be dominated by vertices in A_{left}.

Algorithm 3 An FPT algorithm for t-PATH EDGE DOMINATING SET

Input: $(\mathcal{I} = (G = (V, E), k))$
Parameter: $k + t$
Output: YES if there exists a set $\mathcal{P}$ such that $|\mathcal{P}| \leq k$, each $p \in \mathcal{P}$ is a path of length at most t and the paths in $\mathcal{P}$ dominate every edge in E, else NO.

1: Find a vertex cover A of size at most kt using Lemma 1.
2: If a vertex cover of size kt doesn't exist, then **return** NO instance.
3: Let $I = V \setminus A$.
4: **for** each subset $A_{\text{used}} \subseteq A$ of size at most kt **do**
5: Let $A_{\text{left}} = A \setminus A_{\text{used}}$, $N_{\text{left}} = N(A_{\text{left}}) \cap I$ and $B = I \setminus N_{\text{left}}$.
6: Let $Z = A_{\text{used}} \cup N_{\text{left}}$.
7: If $G[A_{\text{left}}]$ is not an independent set or $|Z| > kt$, go to Step 4 and proceed with the next guess of A_{used}.
8: Invoke Algorithm 2 on the instance $(G, k, Z, B, A_{\text{left}})$.
9: If the subroutine returns YES, **return** YES and terminate.
10: Else, go to Step 4 and proceed with the next guess of A_{used}.
11: **end for**
12: **return** NO instance.

Given this structure of a graph $(A_{\text{used}}, A_{\text{left}}, N_{\text{left}}, B)$, where $B = I \setminus N_{\text{left}}$, solving t-PATH EDS on
(G, k) reduces to solving t-PATH COVER on $(G, k, Z, B, A_{\text{left}})$, where $Z = A_{\text{used}} \cup N_{\text{left}}$. The correctness holds due to Claim 8 listed below.

Claim 8 *A solution to the instance (G, k, Z, B, A_{left}) of t-PATH COVER, where $Z = A_{used} \cup N_{left}$, is also a solution to the instance (G, k) of t-PATH EDS.*

Proof. Let $\mathcal{P}_{\text{cov}}$ be the solution to the instance $(G, k, Z, B, A_{\text{left}})$ of t-PATH COVER. Then, $\mathcal{P}_{\text{cov}}$ consists of at most k paths, each of length at most t, that cover all the vertices in Z and do not contain any vertex from A_{left}. Edges within $G[A_{\text{used}}]$, edges going across A_{used} and A_{left}, edges going across A_{used} and B all contain vertices of A_{used} and hence, have a non-empty intersection with the paths in $\mathcal{P}_{\text{cov}}$. Similarly, edges going across A_{left} and N_{left} contain the vertices of N_{left} and edges going across A_{used} and N_{left} have used vertices as both their endpoints. Thus, these kinds of edges also have a non-empty intersection with the paths in $\mathcal{P}_{\text{cov}}$. As every edge in the graph has a non-empty intersection with the paths in $\mathcal{P}_{\text{cov}}$, the paths in $\mathcal{P}_{\text{cov}}$ dominate every edge in the graph, and hence, the claim holds. □

Since Algorithm 2 is invoked as a subroutine, we have that Algorithm 3 also returns YES with probability at least $\left(1 - \frac{1}{e^c}\right)$. If the given instance is a NO instance, Algorithm 2 always returns NO, and hence, Algorithm 3 also returns NO.

Running time: Computing a vertex cover A of size at most kt takes time $\mathcal{O}^*(1.25284^{kt})$. There are at most 2^{kt} choices of A_{used} and for each choice of A_{used}, Steps 5 and 7 take polynomial time and invoking Algorithm 2 takes $2^{2(kt+t)} \cdot e^{kt}$.

$\mathcal{O}(k^4t^2) \cdot n^{\mathcal{O}(1)}$ time. So, the total time taken is $\mathcal{O}(1.25284^{kt}) \cdot n^{\mathcal{O}(1)} + 2^{2(kt+t)} \cdot (2e)^{kt} \cdot \mathcal{O}(k^4t^2) \cdot n^{\mathcal{O}(1)}$. Thus, the algorithm runs in time $(8e)^{kt}4^tk^4t^2n^{O(1)}$. □

Remark 1. Notice that, the way the algorithm is presented, it returns YES with probability at least $\left(1 - \frac{1}{e^c}\right)$ if the given instance is a YES instance and NO when the given instance is a NO instance. We can tweak Lemma 3 to return a set of colorful paths using the standard technique of backlinks. The necessary modifications in all the algorithms will give us a set of solution paths.

We can derandomize the algorithm using an (n, k)-perfect hash family. An (n, k)-perfect hash family $\mathcal{F}$ is a family of functions from $[n]$ to k such that for every set $S \subseteq [n]$ of size k, there exists a function $f \in \mathcal{F}$ that *splits* S *evenly*. That is, for every $1 \leq j, j' \leq k$, $|f^{-1}(j) \cap S|$ and $|f^{-1}(j') \cap S|$ differ by at most 1. For any $n, k \geq 1$, one can construct an (n, k)-perfect hash family of size $e^k k^{\mathcal{O}(logk)} logn$ in time $e^k k^{\mathcal{O}(logk)} nlogn$ [5,23].

The deterministic algorithm does the following. It finds a vertex cover A of size at most kt, guesses a subset A_{used} of used vertices from A and partitions the vertices of the graph into $Z \uplus B \uplus A_{\text{left}}$, as described in Algorithm 3. If $|Z| \leq kt$, then for each $r \in [0, kt - |A_{\text{used}}|]$, it constructs an $(|Z|, |A_{\text{used}}| + r)$-perfect hash family $\mathcal{F}_r$ and for each $f \in \mathcal{F}_r$, invokes Algorithm 1 on $(G, k, Z, [|A_{\text{used}}| + r], f)$. The properties of an (n, k)-perfect hash family ensure that if there exists a set $\mathcal{P}$ of at most k paths, each of length at most t, in $G[Z \cup B]$ covering all the vertices in Z, then there is a function $f \in \mathcal{F}_r$ that is injective on $V_{\mathcal{P}}$ and consequently, Algorithm 1 finds a set of at most k colorful paths with the required properties for the coloring f. A vertex cover A of size at most kt can be computed in $\mathcal{O}(1.25284^{kt}) \cdot n^{\mathcal{O}(1)}$ time. There are 2^{kt} choices for A_{used}. For each choice of A_{used}, partitioning the vertices into $Z \uplus B \uplus A_{\text{left}}$ takes polynomial time and for each $r \in [kt - |A_{\text{used}}|]$, constructing a $(|Z|, |A_{\text{used}}| + r)$-perfect hash family $\mathcal{F}_r$ takes $e^{kt}(kt)^{\mathcal{O}(logkt)} ktlog(kt)$ time and invoking Algorithm 1 for each $f \in \mathcal{F}_r$ takes $e^{kt}(kt)^{\mathcal{O}(logkt)} log(kt) \cdot 2^{2(kt+t)} \mathcal{O}(k^3t) n^{\mathcal{O}(1)}$. So, the total time taken is $(8e)^{kt}4^t \cdot 2^{\mathcal{O}(log^2kt)} n^{\mathcal{O}(1)}$.

Theorem 4. *Given an instance $\mathcal{I} = (G = (V, E), k)$ of* t-PATH EDGE DOMINATING SET*, there is a deterministic* FPT *algorithm that runs in time $(8e)^{kt}4^t2^{\mathcal{O}(log^2kt)} n^{\mathcal{O}(1)}$ and finds at most k paths, each of length at most t, that dominate all the edges in E.*

5 Conclusion

In this paper, we study two generalizations of EDS from the perspective of parameterized complexity. In particular, we study the EDS problem in hypergraphs, called d-HEDS. We study another extension of EDS where we want to dominate the edges by paths of length t. We give FPT algorithms and polynomial kernels for both problems. We also demonstrate the hardness of the HYPEREDGE DOMINATING SET problem when each element has bounded frequency and any pair of sets in the family intersect in at most one element.

References

1. Alon, N., Yuster, R., Zwick, U.: Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs. In: Leighton, F.T., Goodrich, M.T. (eds.) Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada. pp. 326–335. ACM (1994). https://doi.org/10.1145/195058.195179, https://doi.org/10.1145/195058.195179
2. van Bevern, R.: Towards optimal and expressive kernelization for d-hitting set. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) Computing and Combinatorics - 18th Annual International Conference, COCOON 2012, Sydney, Australia, August 20-22, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7434, pp. 121–132. Springer (2012). https://doi.org/10.1007/978-3-642-32241-9_11, https://doi.org/10.1007/978-3-642-32241-9_11
3. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion-exclusion. SIAM J. Comput. **39**(2), 546–563 (2009). https://doi.org/10.1137/070683933, https://doi.org/10.1137/070683933
4. Chen, J., Kanj, I.A., Xia, G.: Improved parameterized upper bounds for vertex cover. In: Kralovic, R., Urzyczyn, P. (eds.) Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4162, pp. 238–249. Springer (2006). https://doi.org/10.1007/11821069_21, https://doi.org/10.1007/11821069_21
5. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015). https://doi.org/10.1007/978-3-319-21275-3, https://doi.org/10.1007/978-3-319-21275-3
6. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness II: on completeness for W[1]. Theor. Comput. Sci. **141**(1&2), 109–131 (1995). https://doi.org/10.1016/0304-3975(94)00097-3, https://doi.org/10.1016/0304-3975(94)00097-3
7. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science, Springer (1999). https://doi.org/10.1007/978-1-4612-0515-9, https://doi.org/10.1007/978-1-4612-0515-9
8. Escoffier, B., Monnot, J., Paschos, V.T., Xiao, M.: New results on polynomial inapproximability and fixed parameter approximability of edge dominating set. In: Thilikos, D.M., Woeginger, G.J. (eds.) Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7535, pp. 25–36. Springer (2012). https://doi.org/10.1007/978-3-642-33293-7_5, https://doi.org/10.1007/978-3-642-33293-7_5
9. Feige, U.: A threshold of ln n for approximating set cover. J. ACM **45**(4), 634–652 (1998). https://doi.org/10.1145/285055.285059, https://doi.org/10.1145/285055.285059
10. Fernau, H.: edge dominating set: Efficient enumeration-based exact algorithms. In: Bodlaender, H.L., Langston, M.A. (eds.) Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4169, pp. 142–153. Springer (2006). https://doi.org/10.1007/11847250_13, https://doi.org/10.1007/11847250_13

11. Fomin, F.V., Gaspers, S., Saurabh, S., Stepanov, A.A.: On two techniques of combining branching and treewidth. Algorithmica **54**(2), 181–207 (2009). https://doi.org/10.1007/S00453-007-9133-3, https://doi.org/10.1007/s00453-007-9133-3
12. Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.: Efficient computation of representative families with applications in parameterized and exact algorithms. J. ACM **63**(4), 29:1–29:60 (2016). https://doi.org/10.1145/2886094, https://doi.org/10.1145/2886094
13. Fujito, T., Nagamochi, H.: A 2-approximation algorithm for the minimum weight edge dominating set problem. Discret. Appl. Math. **118**(3), 199–207 (2002). https://doi.org/10.1016/S0166-218X(00)00383-8, https://doi.org/10.1016/S0166-218X(00)00383-8
14. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
15. Hagerup, T.: Kernels for edge dominating set: Simpler or smaller. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7464, pp. 491–502. Springer (2012). https://doi.org/10.1007/978-3-642-32589-2_44, https://doi.org/10.1007/978-3-642-32589-2_44
16. Harris, D.G., Narayanaswamy, N.S.: A faster algorithm for vertex cover parameterized by solution size. CoRR **abs/2205.08022** (2022). https://doi.org/10.48550/ARXIV.2205.08022, https://doi.org/10.48550/arXiv.2205.08022
17. Harris, D.G., Narayanaswamy, N.S.: A faster algorithm for vertex cover parameterized by solution size. In: Beyersdorff, O., Kanté, M.M., Kupferman, O., Lokshtanov, D. (eds.) 41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand, France. LIPIcs, vol. 289, pp. 40:1–40:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024). https://doi.org/10.4230/LIPICS.STACS.2024.40, https://doi.org/10.4230/LIPIcs.STACS.2024.40
18. Iwaide, K., Nagamochi, H.: An improved algorithm for parameterized edge dominating set problem. In: Rahman, M.S., Tomita, E. (eds.) WALCOM: Algorithms and Computation - 9th International Workshop, WALCOM 2015, Dhaka, Bangladesh, February 26-28, 2015. Proceedings. Lecture Notes in Computer Science, vol. 8973, pp. 234–245. Springer (2015). https://doi.org/10.1007/978-3-319-15612-5_21, https://doi.org/10.1007/978-3-319-15612-5_21
19. Johnson, D.S.: Approximation algorithms for combinatorial problems. In: Aho, A.V., Borodin, A., Constable, R.L., Floyd, R.W., Harrison, M.A., Karp, R.M., Strong, H.R. (eds.) Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA. pp. 38–49. ACM (1973). https://doi.org/10.1145/800125.804034, https://doi.org/10.1145/800125.804034
20. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA. pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972). https://doi.org/10.1007/978-1-4684-2001-2_9, https://doi.org/10.1007/978-1-4684-2001-2_9

21. Lampis, M.: A kernel of order 2 k-c log k for vertex cover. Inf. Process. Lett. **111**(23-24), 1089–1091 (2011). https://doi.org/10.1016/J.IPL.2011.09.003, https://doi.org/10.1016/j.ipl.2011.09.003
22. Lin, B., Ren, X., Sun, Y., Wang, X.: Constant approximating parameterized k-setcover is w[2]-hard. CoRR **abs/2202.04377** (2022), https://arxiv.org/abs/2202.04377
23. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: 36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995. pp. 182–191. IEEE Computer Society (1995). https://doi.org/10.1109/SFCS.1995.492475, https://doi.org/10.1109/SFCS.1995.492475
24. Niedermeier, R., Rossmanith, P.: An efficient fixed-parameter algorithm for 3-hitting set. J. Discrete Algorithms **1**(1), 89–102 (2003). https://doi.org/10.1016/S1570-8667(03)00009-1, https://doi.org/10.1016/S1570-8667(03)00009-1
25. Norman, R.Z., Rabin, M.O.: An algorithm for a minimum cover of a graph (1959), https://api.semanticscholar.org/CorpusID:120383003
26. Philip, G., Raman, V., Sikdar, S.: Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. ACM Trans. Algorithms **9**(1), 11:1–11:23 (2012). https://doi.org/10.1145/2390176.2390187, https://doi.org/10.1145/2390176.2390187
27. Raman, V., Saurabh, S., Sikdar, S.: Efficient exact algorithms through enumerating maximal independent sets and other techniques. Theory Comput. Syst. **41**(3), 563–587 (2007). https://doi.org/10.1007/S00224-007-1334-2, https://doi.org/10.1007/s00224-007-1334-2
28. van Rooij, J.M.M., Bodlaender, H.L.: Exact algorithms for edge domination. In: Grohe, M., Niedermeier, R. (eds.) Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5018, pp. 214–225. Springer (2008). https://doi.org/10.1007/978-3-540-79723-4_20, https://doi.org/10.1007/978-3-540-79723-4_20
29. Soleimanfallah, A., Yeo, A.: A kernel of order 2k-c for vertex cover. Discret. Math. **311**(10-11), 892–895 (2011). https://doi.org/10.1016/J.DISC.2011.02.014, https://doi.org/10.1016/j.disc.2011.02.014
30. Xiao, M., Kloks, T., Poon, S.: New parameterized algorithms for the edge dominating set problem. In: Murlak, F., Sankowski, P. (eds.) Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6907, pp. 604–615. Springer (2011). https://doi.org/10.1007/978-3-642-22993-0_54, https://doi.org/10.1007/978-3-642-22993-0_54
31. Xiao, M., Nagamochi, H.: A refined exact algorithm for edge dominating set. In: Agrawal, M., Cooper, S.B., Li, A. (eds.) Theory and Applications of Models of Computation - 9th Annual Conference, TAMC 2012, Beijing, China, May 16-21, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7287, pp. 360–372. Springer (2012). https://doi.org/10.1007/978-3-642-29952-0_36, https://doi.org/10.1007/978-3-642-29952-0_36
32. Yannakakis, M., Gavril, F.: Edge dominating sets in graphs. SIAM Journal on Applied Mathematics **38**(3), 364–372 (1980). https://doi.org/10.1137/0138030, https://doi.org/10.1137/0138030