A Fixed-Parameter Algorithm for Max Edge Domination

Tesshu Hanaka and Hirotaka Ono

Department of Economic Engineering, Kyushu University,
Fukuoka 812-8581, Japan

Abstract. In a graph, an edge is said to *dominate* itself and its adjacent edges. Given an undirected and edge-weighted graph $G = (V, E)$ and an integer k, *Max Edge Domination* problem (MaxED) is to find a subset $K \subseteq E$ with cardinality at most k such that total weight of edges dominated by K is maximized. MaxED is NP-hard due to the NP-hardness of the minimum edge dominating set problem. In this paper, we present fixed-parameter algorithms for MaxED with respect to treewidth ω. We first present an $O(\omega^2 \cdot 2^{6\omega+3\omega^2} \cdot k^3 \cdot n)$-time algorithm. We then improve the running time into $O(\omega \cdot 2^{6\omega} \cdot k^3 \cdot n)$, whose exponent is a linear of ω. This improvement enables us to design a subexponential fixed-parameter algorithm of MaxED for apex-minor-free graphs, which is a graph class that includes planar graphs.

Keywords: max edge domination, fixed-parameter algorithm, bounded treewidth, subexponential FPT

1 Introduction

Let $G = (V(G), E(G))$ be an undirected and positive edge-weighted graph, where $V(G)$ is the set of n vertices and $E(G)$ is the set of m edges. These $V(G)$ and $E(G)$ are simply denoted by V and E, respectively. For an edge $e = \{u, v\} \in E$, its weight is denoted by w_e or w_{uv}. For $E' \subseteq E$, we denote by $V(E')$ the set of vertices that appear in E', that is, $V(E') = \bigcup_{e \in E'} e$. An edge is said to *dominate* itself and its all adjacent edges. We denote by $D_G(e)$ the set of edges dominated by an edge e, that is, $D_G(e) = \{e' \in E(G) \mid e' \cap e \neq \emptyset\}$. For a set E' of edges, we denote by $D_G(E')$ the set of edges dominated by an edge in E', that is, $D_G(E) = \{e \in E(G) \mid e \cap V(E') \neq \emptyset\}$. In these notations, we may omit the subscript G if it is clear.

Given $G = (V, E)$ and an integer k, *Max Edge Domination* problem (MaxED) is to find a subset $K \subseteq E$ with cardinality at most k such that total weight of edges dominated by K is maximized. This problem is formulated by the following optimization problem:

$$\max_{K \subseteq E, |K| \leq k} \sum_{e \in D(K)} w_e.$$

In a sense of the decision problem, MaxED for an unweighted graph is equivalent to the well-known *Minimum Edge Dominating Set* (EDS), that is, the problem to find a minimum subset of E' dominating all edges in E. Due to the NP-hardness of EDS, MaxED is NP-hard, and several approximability (or inapproximability) results are known. For example, MaxED is APX-hard [20], and a greedy algorithm achieves approximation ratio $\max\{1-1/e, k/s\}$, where s is the size of maximal matching [16].

In this paper, we consider fixed-parameter tractability of MaxED. Given a problem with input size n and a parameter γ, the problem is said to be *fixed-parameter tractable* (FPT, for short) if it can be solved in $f(\gamma) \cdot n^{O(1)}$ time, where f is a certain function that depends only on parameter γ. An algorithm that achieves the above running time is called a fixed-parameter algorithm. Particularly, if $f(\gamma) = 2^{o(\gamma)}$, the problem is called *subexponential fixed-parameter tractable*. For general concepts of fixed parameter tractability and related topics, see [8, 11, 21]. It is known that EDS is FPT with respect to the solution size [9], but this does not imply the fixed parameter tractability of MaxED with respect to k, because the solution size of EDS can be much larger than k in general. In fact, MaxED with parameter k has shown to be $W[1]$-hard even for unweighted bipartite graphs [15]. This implies that there unlikely exists a fixed-parameter algorithm for MaxED with parameter k.

In this paper, we show that (1) MaxED with respect to treewidth ω is FPT, and (2) MaxED with respect to k is subexponential FPT for apex-minor-free graphs, which is a graph class that includes planar graphs. For the former result, we first present a basic $O(\omega^2 \cdot 2^{6\omega+3\omega^2} \cdot k^3 \cdot n)$-time algorithm for MaxED, and then improve the running time into $O(\omega \cdot 2^{6\omega} \cdot k^3 \cdot n)$. The fixed-parameter tractability of MaxED with respect to treewidth is rather straightforward, but the improved running time plays a key role of the latter result.

There are many combinatorial optimization problems that have subexponential fixed-parameter algorithms for superclasses of planar graphs. A powerful meta-theorem to design a subexponential fixed-parameter algorithm is known for problems having bidimensionality ([4, Theorem 8.1]). Roughly speaking, if a problem has bidimensionality, the treewidth of a planar graph (or a graph in some superclasses of planar graphs) is bounded by $O(\sqrt{k^*})$, where k^* is the optimal value of the problem. By combining this with $2^{O(\omega)}n^{O(1)}$-time algorithm, a subexponential fixed-parameter algorithm can be obtained. Although EDS with respect to solution size is an example of problems having bidimensionality, MaxED with respect to k is unfortunately not. Instead, we try to choose a special K^* among all the optimal solutions. In this strategy, K^* and its neighbors are localized so that the treewidth of the subgraph of G induced by K^* and its neighbors is bounded by $O(\sqrt{k})$. Then, we can expect a similar speeding-up effect. The points become (i) how we localize K^*, and (ii) the design of a fixed-parameter algorithm whose exponent is linear of ω. This scheme is proposed by [13] to design a subexponential fixed-parameter algorithm of *Partial Vertex Cover* with respect to k, which is also not a bidimensional problem, subexponential fixed-parameter algorithms with respect to k for the partial dominating

set and the partial vertex cover of apex-minor-free graphs. Another example of employing this scheme is found in [18]. To apply the scheme, we utilize a generalized version of EDS, say r-EDS, and investigate the approximability. Based on these together with the faster algorithm mentioned in the previous paragraph, we show that there is an algorithm solving MaxED for apex-minor-free graphs in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time.

1.1 Related work

As mentioned above, MaxED is strongly related to EDS. EDS is the problem of finding a minimum subset $S \subseteq E$ such that all edges $e \in E \setminus S$ are adjacent to at least one edge in S. EDS is also known as *Minimum Maximal Matching*. There are many studies for EDS from the viewpoint of (in)approximability, parameterized complexity and exact algorithms. For example, EDS is 2-approximable in polynomial time [14], NP-hard to approximate within any factor better than 7/6 [3], and can be exactly solved in $O^*(1.3160^n)$ time, where O^*-notation suppresses all polynomially bounded factors [23]. EDS is also known to be fixed-parameter tractable with respect to several parameters, e.g., the solution size of EDS, treewidth, and so on. For example, an $O^*(1.821^\tau)$-time algorithm of EDS [22] and an $O^*(2.1479^{k^*})$-time algorithm of EDS for cubic graphs are proposed, where τ is the solution size of the minimum vertex cover, and k^* is the solution size of EDS.

As mentioned before, EDS with solution size is known to be a bidimensional problem. By using the bidimensonality theory, a subexponential fixed-parameter algorithm for apex-minor-free graphs can be designed [5].

Compared with EDS, MaxED itself is less studied. MaxED is a special case of *Maximum Coverage Problem* (MaxC): Given n elements x_i with positive weight w_i, $i = 1, 2, \ldots, n$, sets of $S_1, S_2, \ldots, S_m \subseteq \{x_1, x_2, \ldots, x_n\}$ and a positive integer k, find a set $C \subseteq \{1, 2, \ldots, m\}$ such that $|C| \leq k$ and $\sum_{x_i \in \bigcup_{j \in C} S_j} w_i$ is maximized. Since MaxC is known to be $(1 - 1/e)$-approximable in polynomial time [7, 17], so is MaxED. Though the approximation ratio is tight for MaxC under P$\neq$NP ([10]), MaxED is just known to be APX-hard [20]. As for the parameterized complexity, MaxED with respect to k has been recently shown to be W[1]-hard even for unweighted bipartite graphs [15].

This paper is organized as follows. In Section 2, we introduce notations and definitions. In Sections 3 and 4, we present two fixed-parameter algorithms for MaxED. We first present a basic algorithm in Section 3, and then improve the running time in Section 4. Finally, we show that a $2^{O(\sqrt{k})} \cdot n^{O(1)}$-time algorithm of MaxED for apex-minor-free graphs in Section 5.

2 Preliminaries

Let $G = (V, E)$ be an undirected and edge-weighted graph. For $V' \subseteq V$, let $G[V']$ denote a subgraph of G induced by V'. For $E' \subseteq E$, we simply denote $G[V(E')]$ by $G[E']$.

Here, we introduce the following Lemma 1, whose proof is shown in Appendix.

Lemma 1. There is an optimal solution K^* for MaxED, where K^* forms a matching, that is, every two $e, e' \in K^*$ satisfy $e \cap e' = \emptyset$.

By Lemma 1, we assume that our algorithms for MaxED finds an optimal solution that forms a matching.

2.1 Tree Decomposition

Our algorithms that will be presented in Sections 3 and 4 are based on dynamic programming on tree decomposition. In this subsection, we give the definition of tree decomposition.

A *tree decomposition* of a graph $G = (V, E)$ is defined as a pair $\langle \mathcal{X}, T \rangle$, where $\mathcal{X} = \{X_1, X_2, \ldots, X_N \subseteq V\}$, and T is a tree whose nodes are labeled by $1, 2, \ldots, N$, such that

1. $\bigcup_{i \in I} X_i = V$.
2. For $\forall \{u, v\} \in E$, there exists X_i such that $\{u, v\} \subseteq X_i$.
3. For all $i, j, k \in \{1, 2, \ldots, N\}$, if j lies on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

In the following, we call T a decomposition tree, and we use term "nodes" (not "vertices") for T to avoid a confusion. The width of a tree decomposition $\langle \mathcal{X}, T \rangle$ is is defined by $\max_{i \in \{1,2,\ldots,N\}} |X_i| - 1$, and the treewidth of G, denoted by $\mathbf{tw}(G)$, is the minimum width over all tree decompositions of G. We sometimes use ω to represent $\mathbf{tw}(G)$.

In general, computing $\mathbf{tw}(G)$ of a given G is NP-hard [1], but fixed-parameter tractable with respect to the treewidth [2]. In the following, we assume that a decomposition tree with the minimum treewidth is given.

2.2 r-Edge Dominating Set

We define a new problem by extending the notion of domination. We first define *distance* between two edges $e_1 = \{u_1, v_1\}$ and $e_2 = \{u_2, v_2\}$ as the shortest path length among (u_1, u_2)-path, (u_1, v_2)-path, (v_1, u_2)-path and (v_1, v_2)-path, which we denote by $d(e_1, e_2)$. *r-Edge Dominating Set* (r-EDS) is the problem of finding an edge set $S \subseteq E$ with minimum size such that for every $e \in E \setminus S$, $d(e, e') < r$ holds for some edge $e' \in S$. This problem is clearly a generalization of EDS, because 1-EDS is equivalent to EDS. To design a subexponential fixed-parameter algorithm in Section 5, we design a constant-factor approximation algorithm for 2-EDS.

3 Basic Algorithm Based On Dynamic Programming

In this section, we present a dynamic programming (DP) algorithm based on a decomposition tree. By the assumption above, we are already given a decomposition tree with treewidth of ω. We assume that a decomposition tree T is a rooted binary tree without loss of generality. The algorithm first prepares $k+1$ DP tables for each X_i, so $(k+1) \cdot N$ tables in total. The algorithm runs in the bottom-up manner; it fills tables from leaf nodes to the root node. For simplicity, we assume that the indices of X_i correspond to the order that the algorithm visits X_i; the algorithm fills tables of $X_1, X_2, \ldots, X_N$ in this order.

We further give several assumptions for the tree decomposition. We define a mapping g from E to X_i. For an edge $e \in E$, there exists at least one bag X_i such that $e \subseteq X_i$ by the definition of tree decomposition. If $|\{X_i \in \mathcal{X} \mid X_i \cap e \neq \emptyset\}| = 1$, we define $g(e) = X_i$ such that $X_i \cap e \neq \emptyset$. If $|\{X_i \in \mathcal{X} \mid X_i \cap e \neq \emptyset\}| > 1$, we define $g(e) = X_i$ where i is the smallest index such that $X_i \cap e \neq \emptyset$. By defining g, we make clear in which node we handle e. Based on g, we partition E into $E_1, E_2, \ldots, E_N$, where $E_i = \{e \in E \mid g(e) = X_i\}$. We then define a subgraph $G_i = (V_i, E_i)$ of G, where $V_i = X_i$.

3.1 DP table and its computation

In the basic algorithm, we prepare DP table $A_i^{(r)}$, $r = 0, 1, 2, \cdots, k$, for a bag X_i. Here, r represents the number of edges selected as a part of a solution at the moment. See Table 1 as an example of $A_i^{(r)}$. In the table, $|V_i| = n_i$ and $|E_i| = m_i$. Table $A_i^{(r)}$ consists of $n_i + m_i + 1$ columns and $4^{n_i} \cdot 2^{m_i}$ rows. The first $n_i + m_i$ columns represent the statuses of vertices and edges in G_i. The last column represents the evaluation value ξ of the corresponding statuses. We call the $n_i + m_i$ statuses of a row a *status vector*.

Table 1. $A_i^{(r)}$

v_{i_1}	v_{i_2}	$\cdots$	$v_{i_{n_i}}$	e_{i_1}	e_{i_2}	$\cdots$	$e_{i_{m_i}}$	ξ
0	0	$\cdots$	0	**N**	**N**	$\cdots$	**N**	10
1	0	$\cdots$	0	**N**	**N**	$\cdots$	**N**	11
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$			$\vdots$
1	1	$\cdots$	1	**Y**	**Y**	$\cdots$	**Y**	25
2	0	$\cdots$	0	**N**	**N**	$\cdots$	**N**	13
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
2	2	$\cdots$	2	**Y**	**Y**	$\cdots$	**Y**	25
3	0	$\cdots$	0	**N**	**N**	$\cdots$	**N**	13
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
3	3	$\cdots$	3	**Y**	**Y**	$\cdots$	**Y**	25

Now we explain the statuses of vertices and edges. As a status of vertex v_j, we define four statuses **FUT**, **CUR**, **PAST** and **NULL**. Status **FUT** represents

that v_j does not touch any edge that is or was selected as a part of solution, but touches an edge that will be selected as a part of solution. A vertex of status **FUT** should appear in a parent node. Note that the root node cannot take status **FUT**. Status **CUR** represents that v_j touches an edge in G_i that is selected as a part of solution. Status **PAST** represents that v_j touches an edge that was selected as apart of solution in its offspring node. The remaining vertices take status **NULL**. In a DP table $A_i^{(r)}$, we use $3, 2, 1, 0$ instead of **FUT**, **CUR**, **PAST** and **NULL**, respectively.

For edge statuses, we define two statuses **Y** and **N**, where **Y** represents that the edge is selected as a part of solution and **N** represents that the edge is not selected as a part of solution. Recall that each edge appears in exactly one G_i. Therefore, the vertex status **CUR** and the edge status **Y** are correspond.

Let j_L and j_R be two children nodes of i on T. We define the evaluation value of status vector $\mathbf{p}$ as follows:

$$\xi(A_i^{(r)}(\mathbf{p})) = \sum_{\substack{u \in \{v_i \mid v_i \neq 0\} \\ \{u,v\} \in E_i}} w_{uv} + \max_{s+t+|K_i(\mathbf{p})|=r} \{\xi(A_{j_L}^{(s)}(\mathbf{p}^L)) + \xi(A_{j_R}^{(t)}(\mathbf{p}^R))\}, \quad (1)$$

where $\mathbf{p}^L$ (resp., $\mathbf{p}^R$) is a status vector of $A_{j_L}^{(s)}$ (resp., $A_{j_R}^{(s)}$) that is consistent to $\mathbf{p}$, and $K_i(\mathbf{p})$ is the set of edges in E_i whose status is **Y**. In (1), s and t are the numbers of edges selected as a part of solutions in the offsprings of X_{j_R} and X_{j_L} respectively. The first term represents the edge weight dominated in X_i under $\mathbf{p}$. The second term represents the maximum sum of the evaluation values derived from two children nodes. After filling up the tables of root node, we can obtain an optimal value, and also can obtain an optimal solution by reversely traversing the tables.

Probably a bit tricky part of the algorithm is to define the status **FUT**. This status is introduced to avoid overlooking of the weight of edges that appear in an offspring node. This should be considered because we use the partition $(E_1, E_2, \ldots, E_N)$ of E.

Another important notion is *consistency* of $\mathbf{p}, \mathbf{p}^L$ and $\mathbf{p}^R$. We call $\mathbf{p}$ is inconsistent if it contains impossible statuses. For example, for $e = \{u, v\} \in E_i$, $\mathbf{p}$ is inconsistent if both of the statuses of u and v are **NULL** and $e \in K_i(\mathbf{p})$. For $\mathbf{p}^L$ and $\mathbf{p}^R$, we need to pay an attention for $v \in V_{J_L} \cap V_{j_R}$. Let α_L and α_R be the statuses of v in $\mathbf{p}^L$ and $\mathbf{p}^R$, respectively. Then, the following gives all possible combinations:

$$(\alpha_R, \alpha_L) \in \{\ (\mathbf{NULL}, \mathbf{NULL}), (\mathbf{FUT}, \mathbf{FUT}), (\mathbf{FUT}, \mathbf{CUR}), (\mathbf{FUT}, \mathbf{PAST}), \\ (\mathbf{CUR}, \mathbf{FUT}), (\mathbf{PAST}, \mathbf{FUT})\}.$$

The cases $(\alpha_R, \alpha_L) = (\mathbf{CUR}, \mathbf{CUR}), (\mathbf{CUR}, \mathbf{PAST}), (\mathbf{PAST}, \mathbf{CUR}), (\mathbf{PAST}, \mathbf{PAST})$ can be excluded, because in these combinations solutions do not form a matching. Moreover, we have to consider *consistency* of the parent node and the children nodes. For example, if v's statuses in j_L and j_R are respectively **FUT** and **CUR**, v's status of i should be **PAST**, because v is selected at j_R (i.e., it is a past of i). Let α be the status of $\mathbf{p}$ (i.e., at node i). For each (α_R, α_L) above,

we see possible statuses of α by thinking its chronological order. We then have the following:

- For $(\alpha_R, \alpha_L) = (\mathbf{NULL}, \mathbf{NULL})$, α is **NULL**.
- For $(\alpha_R, \alpha_L) = (\mathbf{FUT}, \mathbf{FUT})$, α is **FUT** or **CUR**.
- For $(\alpha_R, \alpha_L) = (\mathbf{FUT}, \mathbf{CUR})$ or $(\mathbf{CUR}, \mathbf{FUT})$, α is **PAST**.
- For $(\alpha_R, \alpha_L) = (\mathbf{FUT}, \mathbf{PAST})$ or $(\mathbf{PAST}, \mathbf{FUT})$, α is **PAST**.

Overall, we present the following algorithm.

Basic Algorithm
Step 0. Let $i := 1$.
Step 1. Initialize $A_i^{(r)}$ for $r = 0, \ldots, k$.
- Delete rows such that vertex statuses are not consistent to edge statuses.
- Delete rows such that there is a vertex where $\alpha = \mathbf{FUT}$ in $A_i^{(r)}$, while there is not in the parent node of X_i.

Step 2. For every $\mathbf{p}$, compute (1)
Step 3. If $i = N$, output $\max_{r,\mathbf{p}} \xi(A_N^{(r)}(\mathbf{p}))$. Otherwise, let $i := i+1$ and return to Step 1.

We consider the running time of the basic algorithm. The running time of Step 1 is $O(4^\omega \cdot 2^{\omega^2} \cdot k \cdot \omega^2)$ time, because $n_i = O(\omega)$, $m_i = O(\omega^2)$ and the number of rows is $O(4^\omega \cdot 2^{\omega^2} \cdot (k+1))$.

In Step 2, we check all the combinations $A_{j_L}^{(s)}$, $A_{j_R}^{(t)}$ and $A_i^{(r)}$ for $s = 0, 1, \ldots, k$, $t = 0, 1, \ldots, k$, and $r = 0, 1, \ldots, k$. This takes $O(4^{3\omega} \cdot 2^{3\omega^2} \cdot k^3 \cdot \omega^2)$-time in total. Since the procedure of Steps 2 , 3 and 4 is repeated N times, total running time of this algorithm is $O(4^{3\omega} \cdot 2^{3\omega^2} \cdot k^3 \cdot \omega^2 \cdot n)$ time. As a result, we obtain the following theorem.

Theorem 1. *There is an $O(\omega^2 \cdot 2^{6\omega+3\omega^2} \cdot k^3 \cdot n)$-time algorithm for* MaxED.

4 Improvement of the Running time

4.1 Improved Algorithm

We improve the basic algorithm. For the improved algorithm, we prepare a DP table as a new table instead of Table 1. A new DP table is obtained from the corresponding old one by deleting the edge statuses. The size of this DP table is 4^{n_i}. Notice that the size of an improved DP table is much smaller than a basic one. Due to the lack of information, it might be difficult to keep an explicit solution, but it does not matter because we can still compute the evaluation value. In this setting, what we need to consider is to check the consistency. In the basic algorithm, we can easily check the consistency of $\mathbf{p}$ by referring the status vector, but it is not possible for this case.

Actually, this case is also possible to check the consistency. In a new $\mathbf{p}$, all the vertices whose statuses are **CUR**, say $V_i(\mathbf{p})$, should touch an edge selected as a

part of solution in G_i. Since we assume that the selected edges form a matching by Lemma 1, each vertex in $V_i(\mathbf{p})$ touches exactly one edge selected as apart of solution in G_i. In other word, the selected edges form a perfect matching of $G_i[V_i(\mathbf{p})]$. This is clearly a necessary and sufficient condition of the consistency of $\mathbf{p}$, and it can be checked in polynomial time since the maximum matching problem can be solved in polynomial time [19].

Based on the observation, we can rewrite the evaluation value as follows:

$$\xi(A_i^{(r)}(\mathbf{p})) = \sum_{\substack{u\in\{v_i|v_i\neq 0\}\\(u,v)\in E_i}} w_{uv} + \max_{s+t+\frac{|V_i(\mathbf{p})|}{2}=r} \{\xi(A_{j_L}^{(s)}(\mathbf{p}^L)) + \xi(A_{j_R}^{(t)}(\mathbf{p}^R))\},$$

where $\mathbf{p}^L$ (resp., $\mathbf{p}^R$) is a status vector of $A_{j_L}^{(s)}$ (resp., $A_{j_R}^{(s)}$) that is consistent to $\mathbf{p}$. Note that the scheme of the improved algorithm is same as the one of the basic algorithm. Only the difference is how we check the inconsistency of $\mathbf{p}$.

4.2 The Running time of Improved Algorithm

We explain the running time of the improved algorithm. Since the scheme of the basic and improved algorithms are the same, we can easily amount the running time. In Step 1, we should delete the rows whose status vectors are inconsistent. This can be done by applying a maximum matching algorithm, whose running time is $O(\omega^{2.5})$ [19]. Other parts are similarly analyzed, and the total running time of the improved algorithm is $O(4^{3\omega} \cdot k^3 \cdot \omega \cdot n)$-time.

Theorem 2. *There is an $O(\omega \cdot 2^{6\omega} \cdot k^3 \cdot n)$-time algorithm for* MaxED.

5 Subexponentioal fixed-parameter Algorithm

In this section, we will show the following theorem by presenting a subexponential fixed-parameter algorithm.

Theorem 3. *There exists a $2^{O(\sqrt{k})} \cdot n^{O(1)}$-time algorithm for* MaxED *on apex-minor free graphs.*

Let G be an apex-minor-free graph. If $\mathbf{tw}(G) = O(\sqrt{k})$ holds, then Theorem 2 proves Theorem 3. Otherwise we will remove a set I of *irrelevant* edges from G so that at least one optimal solution is a subset of $E \setminus I$ and optimal also for the problem in $G[E \setminus I]$, and we have $\mathbf{tw}(G[E \setminus I]) = O(\sqrt{k})$. Then, applying Theorem 2 to $G[E \setminus I]$, we obtain Theorem 3. To identify such a set I of irrelevant edges, we introduce the notion of *lexicographically smallest solution*. The ideas follow from the ones given by Fomin et al. [13] as mentioned in Introduction.

Definition 1. *Given an ordering $\sigma = e_1 e_2 \dots e_m$ of E and subsets X and Y of E, we say that X is lexicographically smaller than Y, denoted by $X \leq_\sigma Y$, if X is lexicographically smaller than Y, if $E_\sigma^i \cap X = E_\sigma^i \cap Y$ and $e_{i+1} \in X \setminus Y$ for some $i \in \{0, 1, \dots, m\}$, where $E_\sigma^i = \{e_1, e_2, \dots, e_i\}$ for $i \in \{1, 2, \dots, m\}$ and $E_\sigma^0 = \emptyset$. We call a set $K \subseteq E$ the lexicographically smallest (optimal) solution for* MaxED *if for any other solution K' for the* MaxED *we have that $K \leq_\sigma K'$.*

Let $\sigma = e_1 e_2 \dots e_m$ be an ordering of the edges according to the total weight of the edges dominated by an edge in non-increasing order. For $e \in E$, let $\mu(e) = \sum_{e' \in D(e)} w_{e'}$. In the ordering σ,

$$\mu(e_1) \geq \mu(e_2) \geq \cdots \geq \mu(e_{m-1}) \geq \mu(e_m),$$

holds. Throughout the section, we assume that E is ordered by σ, and may use E_σ instead of E to emphasize this. We also denote $\{e_1, e_2, \dots, e_i\}$ by E^i_σ. We will propose an algorithm that finds not an optimal solution but the lexicographically smallest optimal solution for MaxED, which can make it clear to define a set of irrelevant edges. To this end, we give the following three lemmas, though the proof of Lemma 4 is omitted.

Lemma 2. *Given a graph $G = (V, E_\sigma)$, let $K = \{u_{i_1}, u_{i_2}, \dots, u_{i_k}\}$be the lexicographically smallest solution for* MaxED*, where $u_{i_k} = e_j$ for some j. Then, K is a* 2-EDS *of size at most k for $G[E^j_\sigma]$.*

Proof. Show this by contradiction. Assume that a lexicographically smallest solution K of MaxED is not a 2-EDS for $G[E^j_\sigma]$. This implies that there exists an edge e_i $(1 \leq i \leq j)$ such that $D_2(e_i) \cap K = \emptyset$. Let $K' = K \setminus \{e_j\} \cup \{e_i\}$. Clearly, $|K'| = |K|$. Since any edge $e \in D(e_i)$ is not dominated by K, we have

$$\mu(K') \geq \mu(K) - \mu(e_j) + \mu(e_i) \geq \mu(K),$$

a contradiction. □

Lemma 3. *Let G be an apex-minor-free graph. If G has an r-*EDS *of size at most k, $\mathbf{tw}(G) = O(r\sqrt{k})$.*

Proof. If G has an r-EDS of size k, then it has $(2k, r)$-center. Therefore, according to Lemma 8 of [18], the treewidth of G is $O(r\sqrt{k})$. □

Lemma 4. *On apex-minor-free graphs, there exists an EPTAS for* r-EDS*.*

Now we are ready to give a subexponential fixed-parameter algorithm. First, we sort $e_1, e_2, \dots, e_m \in E_\sigma$ and scan it from e_m to e_1. We put a stick in the right of e_m and let $s := m$. In an intermediate stage, if $G[E^j_\sigma]$ does not have a 2-edge dominating set of size at most $(1+\epsilon)k$, let $s := j-1$, $N := N \cup \{e_j\}$, and then we move the stick to the left of e_j. Notice that the edges in the left of the stick belong to $E \setminus N$ and the edges in the right are in N. The contraposition of Lemma 2 denotes that the lexicographically smallest solution for MaxED K lies $E \setminus N$, that is, $K \subseteq E \setminus N$. If $G[E^j_\sigma]$ has a 2-edge dominating set of size at most $(1+\epsilon)k$, then we find a subgraph G' such that $\mathbf{tw}(G') = O(\sqrt{k})$ and there exists $K' \subseteq E(G')$ satisfying $\mu(K) = \mu(K')$ for an optimal solution K of G, where $|K'| \leq k$ and $|K| \leq k$.

Given the parameter $(G = (V, E_\sigma), k, \epsilon, \emptyset)$ where $\epsilon > 0$, the algorithm is described as follows.

Subexponential fixed-parameter Algorithm

Step 0. Let $p := m$

Step 1. While there does not exist 2-edge dominating set of size at most $(1+\epsilon)k$ for $G[E^p_\sigma]$, repeat $N := N \cup \{e_p\}$, $p := p - 1$.

Step 2. Let $I = \{e \mid e \in N, D(e) \subseteq N\}$ and $E' = E \setminus I$.

Step 3. Find a tree decomposition of $G' = G[E']$ using the constant factor approximation algorithm of Demaine et al. [6] for computing the treewidth of H-minor-free graph.

Step 4. Apply the algorithm of Theorem 2 to $G[E']$.

The correctness of the algorithm can be shown by following the proof of Theorem 1 of [13]. In Step 1, we identify an edge set N such that $N \cap K = \emptyset$. Let e_p be a maximum index edge in $E \setminus N$. We check whether $G[E^p_\sigma]$ has 2-edge dominating set of size at most $(1+\epsilon)k$ by Lemma 4. If $G[E^p_\sigma]$ does not have it, then $K = \{u_{i_1}, u_{i_2}, \ldots u_{i_k}\}$ where $u_{i_k} = e_p$ is not the lexicographically smallest solution for MaxED by Lemma 2. Therefore, $e_p \notin K$. We will show letter half is valid. Note that edges in N are not candidates. Thus, an edge $e \in N$ adjacent to only edges in N is not dominated by K, that is, the set I is a set of irrelevant edges. Therefore, we delete a set of such edges as I. Let $E' = E \setminus I$. There exists K of size at most k in G such that $\mu(K) = \max_{K \subseteq E, |K| \leq k} \mu(K)$ if and only if there exists $K' \subseteq E'$ in G' such that $|K'| \leq k$ and $\mu(K') = \max_{K' \subseteq E', |K| \leq k} \mu(K')$. Hence, we will find K' in G' where $|K'| \leq k$ by Theorem 2.

We analyze the running time of this algorithm. When the loop in Step 2. is broken out, $G[E \setminus N]$ has 2-edge dominating set of size at most $(1+\epsilon)k$. Let D_2 be 2-edge dominating set of size at most $(1+\epsilon)k$. Then, D_2 is 3-edge dominating set for $G[E']$ because all edges such that $e \in N \cap E'$ are adjacent to edges in $E \setminus N$. Therefore, $\mathbf{tw}(G') = O(3\sqrt{(1+\epsilon)k}) = O(\sqrt{k})$ is shown by Lemma 3. We use the constant factor approximation algorithm of Demaine et al. [6] to compute the treewidth of H-minor-free graph, then we find tree decomposition such that the size of treewidth is $O(\sqrt{k})$ for $G[E']$ in $n^{O(1)}$-time. Finally, we use the algorithm of Theorem 2 to find optimal solution for MaxED in $O(2^{6\omega} \cdot k^3 \cdot \omega \cdot n)$-time. Therefore, our algorithm achieves running time $2^{O(\sqrt{k})} \cdot n^{O(1)}$.

References

1. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM Journal on Algebraic Discrete Methods 8(2), 277–284 (1987)
2. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on Computing 25(6), 1305–1317 (1996)
3. Chlebík, M., Chlebíková, J.: Approximation hardness of edge dominating set problems. Journal of Combinatorial Optimization 11(3), 279–290 (2006)
4. Demaine, E.D., Hajiaghayi, M.: The bidimensionality theory and its algorithmic applications. The Computer Journal 51(3), 292–302 (2008)
5. Demaine, E.D., Hajiaghayi, M.: Linearity of grid minors in treewidth with applications through bidimensionality. Combinatorica 28(1), 19–36 (2008)

6. Demaine, E.D., Hajiaghayi, M., Kawarabayashi, K.i.: Algorithmic graph minor theory: Decomposition, approximation, and coloring. In: Proceedings of 46th Annual IEEE Symposium on Foundations of Computer Science. pp. 637–646. IEEE (2005)
7. Dobson, G.: Worst-case analysis of greedy heuristics for integer programming with n onnegative data. Mathematics of Operations Research 7(4), 515–531 (1982)
8. Downey, R.G., Fellows, M.R.: Parameterized complexity, vol. 3. Springer-Heidelberg (1999)
9. Escoffier, B., Monnot, J., Paschos, V., Xiao, M.: New results on polynomial inapproximability and fixed parameter approximability of edge dominating set. In: Parameterized and Exact Computation, Lecture Notes in Computer Science, vol. 7535, pp. 25–36. Springer Berlin Heidelberg (2012)
10. Feige, U.: A threshold of ln n for approximating set cover. Journal of the ACM (JACM) 45(4), 634–652 (1998)
11. Flum, J., Grohe, M.: Parameterized complexity theory, vol. 3. Springer (2006)
12. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Bidimensionality and eptas. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 748–759. SIAM (2011)
13. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Subexponential algorithms for partial cover problems. Information Processing Letters 111(16), 814–818 (2011)
14. Fujito, T., Nagamochi, H.: A 2-approximation algorithm for the minimum weight edge dominating set problem. Discrete Applied Mathematics 118(3), 199–207 (2002)
15. Guo, J., Shrestha, Y.: Parameterized complexity of edge interdiction problems. In: Computing and Combinatorics, Lecture Notes in Computer Science, vol. 8591, pp. 166–178. Springer International Publishing (2014)
16. Hanaka, T., Ono, H.: Approximation ratios of greedy algorithms for max edge domination. In: Proceedings of Hinokuni Information Symposium (in Japanese). Information Processing Society of Japan (2013)
17. Hochbaum, D.S.: Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In: Approximation algorithms for NP-hard problems. pp. 94–143. PWS Publishing Co. (1996)
18. Ishii, T., Ono, H., Uno, Y.: Subexponential fixed-parameter algorithms for partial vector domination. In: Combinatorial Optimization, pp. 292–304. Lecture Notes in Computer Science, Springer International Publishing (2014)
19. Micali, S., Vazirani, V.V.: An $O(\sqrt{|V|}|E|)$ algoithm for finding maximum matching in general graphs. In: Proceedings of 21st Annual Symposium on Foundations of Computer Science. pp. 17–27. IEEE (1980)
20. Miyano, E., Ono, H.: Maximum domination problem. In: Proceedings of the Seventeenth Computing: The Australasian Theory Symposium-Volume 119. pp. 55–62. Australian Computer Society, Inc. (2011)
21. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press (2006)
22. Xiao, M., Nagamochi, H.: Parameterized edge dominating set in cubic graphs. In: Frontiers in Algorithmics and Algorithmic Aspects in Information and Management, Lecture Notes in Computer Science, vol. 6681, pp. 100–112. Springer Berlin Heidelberg (2011)
23. Xiao, M., Nagamochi, H.: A refined exact algorithm for edge dominating set. In: Theory and Applications of Models of Computation, Lecture Notes in Computer Science, vol. 7287, pp. 360–372. Springer Berlin Heidelberg (2012)

A APPENDIX

This appendix provides the proofs of the results that have been omitted due to space reasons. They may be read to the discretion of the program committee.

A.1 Proof of Lemma 1

Proof. When $k \geq s$ where s is the size of a minimum maximal matching, a minimum maximal matching is an optimal solution for MaxED.

Thus we assume that $k < s$. Let the set K be an optimal solution for MaxED and we call an edge $e \in K$ optimal edge. Suppose that there is no optimal solution with a matching structure. Then, there exists at least one vertex incident to at least two one optimal edges. We call such a vertex *heavy* vertex and edges composing it *composing* edges. Suppose that *heavy* vertices touches two optimal edges. Notice that there is at least one edge not dominated by K due to $k < s$. Let an edge e^* be one of such uncovered edge and let a vertex v^* be *heavy* vertex closest to e^*. Then, we consider the shortest path from v^* to e^* through *composing* edges. Let the edge e' be a *composing* edge closer to e^* and let a vertex v' be another endpoint of e', that is, $e' = \{v^*, v'\}$. All edges incident to v' are not optimal edges because v^* is *heavy* vertex. closest to e^*. If there is at least one edge incident to v' not dominated by K except for e', we can replace optimal edge e' with it without changing the total weight dominated by K. Then, this optimal solution has a matching structure(*), thus this is contradiction. Otherwise, let $K' = K \setminus \{e'\} \cup \{e^*\}$. Then, K' dominates at least more e' than K. Therefore, this contradicts to the optimality of K.

In the case that more than three optimal edges compose a *heavy* vertex, K does not have a matching structure in (*). However, we can lead a contradiction by considering the shortest path again. □

A.2 Proof of Lemma 4

To show Lemma 4, we use the result of [12].

For a problem Π, let $\phi_\Pi(G, S)$ be the feasibility constraint returning **true** if S is feasible and **false** otherwise. Let $\kappa_\Pi(G, S)$ be objective function. In most case, $\kappa_\Pi(G, S)$ will return $|S|$. We will only consider problems such that every instance has at least one feasible solution. Let $\mathcal{U}$ be the set of all graphs. For a graph optimization problem Π, let $\pi : \mathcal{U} \to \mathbb{N}$ be the function returning the objective function value of optimal solution of Π on G. For a graph G and a partition of $V(G)$ into $L \biguplus S \biguplus R$ such that $N(L) \subseteq S$ and $N(R) \subseteq S$, we define $G_L(G_R)$ as a graph obtained from G by contracting every connected component of $G[R](G[L])$ into the minimum index vertex in S.

Definition 2 ([12]). *A contraction-bidimensional problem has the separation property if given any graph G and a partition of $L \biguplus S \biguplus R$ such that $N(L) \subseteq S$ and $N(R) \subseteq S$, and given an optimal solution OPT to G, $\pi(G_L) \leq \kappa_\Pi(G_L, OPT \setminus E(G[R])) + O(|S|)$ and $\pi(G_R) \leq \kappa_\Pi(G_R, OPT \setminus E(G[L])) + O(|S|)$.*

Definition 3 ([12]). *A graph optimization problem Π with objective function κ_Π is called* reducible *if there exists a* **MIN/MAX-CMSO** *problem Π' and a functionf $: \mathbb{N} \to \mathbb{N}$ such that*

- *there is a polynomial time algorithm that given G and $X \subseteq V(G)$ outputs G' such that $\pi'(G') = \pi(G) \pm O(|X|)$ and $\mathbf{tw}(G') \leq f(\mathbf{tw}(G \setminus X)$.*
- *there is a polynomial time algorithm that given G and $X \subseteq V(G)$, G' and a vertex (edge) set S' such that holds $P_{\Pi'}(G', S')$, outputs S such that $\phi_\Pi(G, S) = \mathbf{true}$ and $\kappa_\Pi(G, S) = |S'| \pm O(|X|)$*

Then, the following lemma has been shown.

Lemma 5 ([12]). *Let Π be a reducible contraction-bidimensional problem with the separation property and H be a (apex) graph. There is an EPTAS for Π on the class of graphs excluding H as a minor.*

Therefore, there exists EPTAS for r-EDS if it is reducible and has separation property.

Proof of Lemma 4. Let $V(G) = L \biguplus S \biguplus R$ such that $N(L) \subseteq S$ and $N(R) \subseteq S$. Let Z be an optimal solution of r-EDS for G and Z_L be an optimal solution of r-EDS for G_L. Moreover, in a graph G_L, let $K^*_L[S]$ be an edge set such that $|K^*_L[S]| \leq |S|$ and for arbitrary non-isolated vertices u and v in S there exists at least one edge such that $\{u, v\} \in K^*_L[S]$. Then, $(Z \setminus E(G[R])) \cup K^*_L[S]$ is r-EDS of G_L because the edges dominated by $e \in E(G[R])$ or $(u, v) \in E(G)$ where $u \in R$ and $v \in S$ are dominated by $K^*_L[S]$. Because Z_L is an optimal solution in G_L,

$$\begin{aligned} |Z_L| &\leq |(Z \setminus E(G[R])) \cup K^*[S]| \\ &\leq |(Z \setminus E(G[R]))| + |K^*[S]| \\ &\leq |(Z \setminus E(G[R]))| + |S|. \end{aligned}$$

Therefore, r-EDS has separation property.

Given a graph G and edge set Y, let X be a vertex set of endpoints of edges in Y. Let $G' = G \setminus X$ and $R = D_r(Y) \setminus Y$. Note that $\mathbf{tw}(G') = \mathbf{tw}(G \setminus X)$. The annotated problem Π' is to find the minimum sized edge set $S' \subseteq E(G')$ such that every edge $e \in E(G) \setminus (S \cup R)$ is at distance at most r from some edges in S. Here, for any r-edge dominating set in G of Π, $S \setminus Y$ is a feasible solution in G' of Π'. Conversely, for any r-edge dominating set in G' of Π', $S' \cup Y$ is a feasible solution in G of Π. Because $\pi'(G') \leq |S \setminus Y| \leq |S| - |Y|$ for any S,

$$\pi'(G') \leq \pi(G) - |Y| \leq \pi(G) - \frac{1}{2}|X|.$$

Also, given G, $Y \subseteq E(G)$ and a set $X \subseteq V(G)$ of endpoints of Y, G' and an edge set S', let $S = S' \cup Y$. Then, considering $|Y| \leq |X| \leq 2|Y|$, they holds:

- $\phi_\Pi(G, S) = \mathbf{true}$
- $\kappa_\Pi(G, S) = |S'| + |Y| = |S'| + O(|X|)$.

Therefore, r-EDS is reducible. □