A Fixed-Parameter Algorithm for Max Edge Domination*

Tesshu Hanaka and Hirotaka Ono

Department of Economic Engineering, Kyushu University,
Fukuoka 812-8581, Japan
ono@csce.kyushu-u.ac.jp

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 $O(3^\omega \cdot k \cdot n \cdot (k+\omega^2))$-time algorithm. This algorithm 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:

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

* This work is partially supported by KAKENHI grant number 24106004, 25104521, 26540005 and Asahi Glass Foundation.

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 [21], and a greedy algorithm achieves approximation ratio $\max\{1 - 1/e, k/s\}$, where s is the size of maximal matching [17].

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 [9, 12, 22]. It is known that EDS is FPT with respect to the solution size [10], 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 [3]. Recently, Guo, J. et al. proved that MaxED is $W[1]$-hard even for unweighted bipartite graphs [16]. 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 present an $O(3^{\omega} \cdot k \cdot n \cdot (k+\omega^2))$-time algorithm for MaxED. 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 ([5, 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 [14] 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 [19]. 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 [15], NP-hard to approximate within any factor better than 7/6 [4], and can be exactly solved in $O^*(1.3160^n)$ time, where O^*-notation suppresses all polynomially bounded factors [24]. 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 [23] 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 [6].

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 [8, 18], so is MaxED. Though the approximation ratio is tight for MaxC under P$\neq$NP ([11]), MaxED is just known to be APX-hard [21]. As for the parameterized complexity, MaxED with respect to k has been shown to be $W[1]$-hard[3]. Recently, Guo et al. proved that MaxED is $W[1]$-hard even for unweighted bipartite graphs [16].

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 $G[E']$.

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.

Definition 1. *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.

Moreover, we introduce a very useful tree decomposition for some algorithms, called *nice tree decomposition*. In the sense, it is a special binary tree decomposition.

Definition 2. *. A tree decomposition $\langle \mathcal{X}, T \rangle$ is called* nice tree decomposition *if it satisfies the following:*

1. *T is rooted at a designated node $X_N \in \mathcal{X}$, called root node.*
2. *Every node of the tree T has at most two children nodes.*
3. *The nodes of T hold one of the following four node types:*
   - *A* leaf *node i which has no children and the corresponding leaf bag X_i has $|X_i| = 1$.*
   - *An* introduce *node i which has one child j with $X_i = X_j \cup \{v\}$ for a vertex $v \in V$.*
   - *A* Forget *node i which has one child j with $X_i = X_j \setminus \{v\}$ for a vertex $v \in V$.*
   - *A* Join *node i which has two children $j, l \in \mathcal{X}$ with $X_i = X_j = X_l$.*

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 Fixed-Parameter Algorithm Bounded Treewidth

In this section, we present a dynamic programming (DP) algorithm based on a nice decomposition tree. By the assumption above, we are already given a nice decomposition tree with treewidth of ω. We assume that 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. We define $g(e) = X_i$ where i is the smallest index such that $e \subseteq X_i$. 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 $V_i = X_i$.

Now, we prepare DP tables $A_i^{(r)}$ like Table 1 for each X_i, where $r = 0, 1, \cdots, k$. Here, r represents the number of edges selected as a part of a solution at the moment. In table $A_i^{(r)}$, let $|V_i| = n_i$. Table $A_i^{(r)}$ consists of $n_i + 1$ columns and 3^{n_i} rows. The first n_i columns represent the statuses of vertices in G_i. The last column represents the value of the corresponding statuses output by the function defined latter. Each vertex v in X_i has the status $c(v) \in \{0, 1, 2\}$ and for each row in $A_i^{(r)}$, we define the *coloring* $c = (c(v_1), c(v_2), \cdots, c(v_{n_i})) \in \{0, 1, 2\}^{|V_i|}$. We also define $c(V_i \backslash V')$ where $V' \subseteq V_i$ as a part of coloring c. Status 0 represents the vertex which is not the endpoint of the solution, while status 1 means that

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

v_1	v_2	$\cdots$	v_{n_i}	$f_i^{(r)}()$
0	0	$\cdots$	0	10
1	0	$\cdots$	0	11
$\vdots$	$\vdots$		$\vdots$	$\vdots$
1	1	$\cdots$	1	-
2	0	$\cdots$	0	-
$\vdots$	$\vdots$		$\vdots$	$\vdots$
2	2	$\cdots$	2	-

the vertex is the endpoint of that. In the sense, status 2 is special status. That is, the vertex with 2 is not the endpoint of the solution in X_i but it will be the endpoint of that after X_i.

We define the function $f_i^{(r)}(c) : \{0,1,2\}^{|V_i|} \to \mathbb{R} \cup \{-\infty\}$ for each DP table $A_i^{(r)}$. This function's value represents the total weight of edges dominated by the part of solution until X_i. If $f_i(c) = -\infty$, it means the coloring c is invalid.

We perform dynamic programming from leaf nodes. First, we start initialization for all leaf node. For each leaf node i, we assume that $X_i = \{x\}$ and $c = c(x)$. Then, we compute $f_i^{(r)}(c)$ as follows:

$$f_i^{(r)}(c) := \begin{cases} 0 \ (r = 0 \text{ and } c \in \{0,2\}) \\ -\infty \ (\text{otherwise}) \end{cases}.$$

Then, we explain update step. Let c' be the coloring in X_{i-1}. Update methods are different for each node type as follows.

Introduce node $(X_i = X_{i-1} \cup \{x\})$:

There are following two cases for an introduce node.

case 1. $c = c' \times \{0\}$ or $c = c' \times \{2\}$ where $c' = c'(V_{i-1}) = c(V_i \setminus \{x\})$.

case 2. There is a neighbor z of x in X_{i-1} such that $c = c(V_i \setminus (\{z\} \cup \{x\})) \times \{1\} \times \{1\}$ and $c' = c(V_{i-1} \setminus \{z\}) \times \{2\}$.

For each case, $f_i^{(r)}(c)$ is defined as follows.

$$f_i^{(r)}(c) := \begin{cases} f_{i-1}^{(r)}(c') + \sigma(c) & (\text{case 1}) \\ f_{i-1}^{(r-1)}(c') + \sigma(c) & (\text{case 2}) \\ -\infty & (\text{otherwise}) \end{cases},$$

where $\sigma(c) = \sum_{\substack{u \in \{v_i \mid c(v_i) \neq 0\} \\ (u,v) \in E_i}} w_{uv}$. The value $\sigma(c)$ represents the total weight of edges dominated by the coloring c in X_i.

Forget node $(X_i = X_{i-1} \setminus \{x\})$:

For a forget node, we can immediately define $f_i^{(r)}(c)$ as follows:

$$f_i^{(r)}(c) := \max\{f_{i-1}^{(r)}(c \times \{0\}), f_{i-1}^{(r)}(c \times \{1\})\}.$$

We do not consider the case that $c(x) = 2$ because x will be never the endpoint of the solution due to forget node.

Join node $(X_i = X_j = X_l)$:

We assume that X_j and X_l are the children nodes of X_i. For a join node, we have to compute $f_i^{(r)}$ for the combination of X_j and X_l. Because $X_i = X_j = X_l$, there is the coloring c in each node X_i, X_j and X_l. Thus, we can define $f_i^{(r)}(c)$ as follows:

$$f_i^{(r)}(c) := \max_{0 \leq r_j \leq r} \{f_j^{(r_j)}(c), f_l^{(r-r_j)}(c)\}.$$

Finally, we compute the root node. It is one of the four node type, thus we firstly update DP table following above methods. Then we modify the value $f_r(c)$. That is, if there is a vertex v such that $c(v) = 2$ in coloring c, let $f_N^{(r)}(c) := -\infty$. Then we output $\max_{r,c} f_N^{(r)}(c)$.

Now, we consider the running time of this algorithm. For each leaf node, we can initialize DP tables in $O(k)$-time since $|X_i| = 1$. Then, we analyze update step. When the node is introduce node, the running time is $O(3^\omega \cdot k \cdot \omega^2)$ because $n_i = O(\omega)$ for each node X_i and we can calculate $\sigma(c)$ in $O(\omega^2)$-time for each row. For a forget node, we only check two coloring $c \times \{0\}$ and $c \times \{1\}$ of X_{i-1} corresponding to c in X_i. Therefore, the running time of a forget node is $O(3^\omega \cdot k)$. In a join node, we search the best combination of X_j and X_l for each $f_i^{(r)}(c)$ in $O(k)$-time. Thus, the running time of a forget node is $O(3^\omega \cdot k^2)$-time. Finally, we modify DP table and output $\max_{r,c} f_N^{(r)}(c)$ in the root node in $O(3^\omega \cdot k \cdot \omega)$-time. Thus, the total running time is as follows:

$$O(k \cdot N) + O(3^\omega \cdot k \cdot N \cdot (k + \omega^2)) + O(3^\omega \cdot k \cdot \omega) = O(3^\omega \cdot k \cdot n \cdot (k + \omega^2)).$$

Therefore, we can show the following theorem.

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

4 Subexponentioal Fixed-Parameter Algorithm

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

Theorem 2. *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 1 proves Theorem 2. 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 1 to $G[E \setminus I]$, we obtain Theorem 2. 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. [14] as mentioned in Introduction.

Definition 3. *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 $E_\sigma^i \cap X = E_\sigma^i \cap Y$ and $e_{i+1} \in Y \setminus X$ 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 $\{e_1, e_2, \ldots, e_i\}$ 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 3 is omitted.

Lemma 1. *Given a graph $G = (V, E_\sigma)$, let $K = \{u_{i_1}, u_{i_2}, \ldots, u_{i_k}\}$be the lexicographically smallest solution for* MaxED*, where $u_{i_k} = e_j$ for some j. Then, K is a* 2-EDS *of size 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 2. *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 [19], the treewidth of G is $O(r\sqrt{k})$. □

Lemma 3. *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 $e_1, e_2, \ldots, e_m \in E_\sigma$ 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 1 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. [7] for computing the treewidth of H-minor-free graph.
Step 4. Apply the algorithm of Theorem 1 to $G[E']$.

The correctness of the algorithm can be shown by following the proof of Theorem 1 of [14]. In Step 1, we identify an edge set N that are not used in lexicographically smallest solution of MaxED. 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 3. If $G[E^p_\sigma]$ does not have it, then $\{u_{i_1}, u_{i_2}, \ldots u_{i_k}\}$ satisfying $u_{i_k} = e_p$ is not the lexicographically smallest solution for MaxED by Lemma 1. Therefore, $e_p \notin K$. We will show latter 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 1.

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 2. We use the constant factor approximation algorithm of Demaine et al. [7] 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 1 to find optimal solution for MaxED in $O(3^\omega \cdot k \cdot n \cdot (k + \omega^2))$-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. Cai, L.: Parameterized Complexity of Cardinality Constrained Optimization Problems. In: The Computer Journal 51(1), 102–121 (2008)
4. Chlebík, M., Chlebíková, J.: Approximation hardness of edge dominating set problems. Journal of Combinatorial Optimization 11(3), 279–290 (2006)
5. Demaine, E.D., Hajiaghayi, M.: The bidimensionality theory and its algorithmic applications. The Computer Journal 51(3), 292–302 (2008)
6. Demaine, E.D., Hajiaghayi, M.: Linearity of grid minors in treewidth with applications through bidimensionality. Combinatorica 28(1), 19–36 (2008)
7. 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)

8. Dobson, G.: Worst-case analysis of greedy heuristics for integer programming with n onnegative data. Mathematics of Operations Research 7(4), 515–531 (1982)
9. Downey, R.G., Fellows, M.R.: Parameterized complexity, vol. 3. Springer-Heidelberg (1999)
10. 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)
11. Feige, U.: A threshold of ln n for approximating set cover. Journal of the ACM (JACM) 45(4), 634–652 (1998)
12. Flum, J., Grohe, M.: Parameterized complexity theory, vol. 3. Springer (2006)
13. 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)
14. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Subexponential algorithms for partial cover problems. Information Processing Letters 111(16), 814–818 (2011)
15. Fujito, T., Nagamochi, H.: A 2-approximation algorithm for the minimum weight edge dominating set problem. Discrete Applied Mathematics 118(3), 199–207 (2002)
16. 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)
17. 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)
18. 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)
19. 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)
20. 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)
21. 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)
22. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press (2006)
23. 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)
24. 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)