Minimum-Cost b-Edge Dominating Sets on Trees

Takehiro Ito[1,a)] Naonori Kakimura[2,b)] Naoyuki Kamiyama[3,c)] Yusuke Kobayashi[2,d)]
Yoshio Okamoto[4,e)]

Abstract: We consider the minimum-cost b-edge dominating set problem. This is a generalization of the edge dominating set problem, but the computational complexity for trees is an astonishing open problem. We make steps toward the resolution of this open problem in the following three directions. (1) We give the first combinatorial polynomial-time algorithm for paths. Prior to our work, the polynomial-time algorithm for paths used linear programming, and it was known that the linear-programming approach could not be extended to trees. Thus, our algorithm would yield an alternative approach to a possible polynomial-time algorithm for trees. (2) We give a fixed-parameter algorithm for trees with the number of leaves as a parameter. Thus, a possible **NP**-hardness proof for trees should make use of trees with unbounded number of leaves. (3) We give a fully polynomial-time approximation scheme for trees. Prior to our work, the best known approximation factor was two. If the problem is **NP**-hard, then a possible proof cannot be done via a gap-preserving reduction from any **APX**-hard problem unless **P** = **NP**.

1. Introduction

Covering problems are very fundamental in the study of graph algorithms. By objects to cover and objects to be covered, the following terms are assigned. When vertices cover vertices, we obtain the minimum dominating set problem; when vertices cover edges, we obtain the minimum vertex cover problem; when edges cover vertices, we obtain the minimum edge cover problem; when edges cover edges, we obtain the minimum edge dominating set problem. While the minimum edge cover problem can be solved in polynomial time, the other three problems are **NP**-hard for general graphs. Nevertheless, they can be solved in linear time for trees.

In some applications, variations of these problems are considered. One variation imposes cost on objects to cover, and another variation imposes demand on objects to be covered. Then, we want to select some objects to cover, possibly multiple times, so that each object to be covered is covered at least as many times as its demand. The objective is to minimize the total cost of selected objects multiplied by the times they are selected. The focus of this paper is the version for the minimum edge dominating set problem.

Formally, we define the minimum-cost b-edge dominating set problem (b-EDS, for short) as follows. In b-EDS, we are given a simple graph G. We denote by VG and EG the sets of vertices and edges of G, respectively. We adopt this notation for any graphs in this paper. In addition, a demand function $b\colon EG \to \mathbb{Z}_+$ and a cost function $c\colon EG \to \mathbb{R}_+$ are given.[*1] For each edge e of G, we denote $\delta(e)$ by the set of all edges sharing end vertices with e, including e itself. For each vertex v of G, we denote by $\delta(v)$ the set of edges containing v. A vector s in $\mathbb{Z}_+^{EG}$ is called a *b-edge dominating vector* of G, if $s(\delta(e)) \geq b(e)$ for every edge e of G.[*2] The value $s(e)$ represents the number of times the edge e is selected. The *cost* of a b-edge dominating vector s of G, denoted by $\mathsf{cost}(s)$, is defined as $\langle c, s\rangle$, where $\langle\cdot,\cdot\rangle$ represents the inner product of two vectors. The goal of b-EDS is to find a b-edge dominating vector with the minimum cost. It is known [13] that b-EDS is **NP**-hard for general graphs even if $b(e) = 1$ and $c(e) = 1$ for every edge e of G. An $8/3$-approximation algorithm for general graphs and a 2-approximation algorithm for bipartite graphs were presented [1]. When $b(e) = 1$ for every edge e of G, 2-approximation algorithms were also proposed [6], [9].

One astonishing open problem on b-EDS is to determine the computational complexity of the problem on trees. It is not known if the problem is **NP**-hard or polynomial-time solvable. If $b(e) = 1$ for every edge e of G or $c(e) = 1$ for every edge e of G, then b-EDS can be solved in polynomial time for trees [2].[*3] Therefore, the combination of arbitrary b and arbitrary c makes the problem troublesome. The best known approximation factor is two, which is a consequence from bipartiteness of trees [1]. Even for paths, no combinatorial polynomial-time algorithm was known while a strongly polynomial-time algorithm can be de-

1 Tohoku University, Japan
2 University of Tokyo, Japan
3 Kyushu University, Japan
4 University of Electro-Communications, Japan
a) takehiro@ecei.tohoku.ac.jp
b) kakimura@global.c.u-tokyo.ac.jp
c) kamiyama@imi.kyushu-u.ac.jp
d) kobayashi@mist.i.u-tokyo.ac.jp
e) okamotoy@uec.ac.jp

[*1] We denote by $\mathbb{Z}$, $\mathbb{Z}_+$, $\mathbb{R}$, and $\mathbb{R}_+$ the sets of integers, nonnegative integers, real numbers, and nonnegative real numbers, respectively.

[*2] For each set X, each vector ξ in $\mathbb{R}^X$, and each subset Y of X, we define $\xi(Y) := \sum_{x\in Y} \xi(x)$. Thus, $s(\delta(e)) = \sum_{e'\in\delta(e)} s(e')$.

[*3] In [2], the authors claimed that b-EDS on trees can be solved in polynomial time via linear programming. However this claim is not correct (see [3]).

signed via linear programming [10], [11].

1.1 Contributions and Techniques

We make steps toward the resolution of the complexity status of b-EDS on trees in the following three directions.

1.1.1 Combinatorial Algorithm for Paths

In Section 2, we give the first combinatorial algorithm for b-EDS on paths which runs in strongly polynomial time. This result would yield an alternative approach to a possible polynomial-time algorithm for trees for the attempt to prove that b-EDS can be solved in polynomial time. A polynomial-time algorithm using linear programming could not be extended to trees because the coefficient matrix of a natural integer-programming formulation of the problem is totally unimodular for paths, but not necessarily so for trees.

To give a combinatorial strongly polynomial-time algorithm for paths, we first give a dynamic-programming algorithm which runs in pseudo-polynomial time. Then, we construct a "compact" representation of the DP table. To construct such a compact representation of the DP table, we find a partition $\mathcal{R}$ of the domain of the DP table so that in each part in $\mathcal{R}$, values of the DP table can be represented by a linear function. We will prove that we can construct such a partition whose size is bounded by a polynomial in the input size, which implies that we can "simulate" our dynamic-programming algorithm in strongly polynomial time. To the best of the authors' knowledge, this technique of compressing the DP table is new, and should be of independent interest.

1.1.2 Fixed-Parameter Algorithm for Trees

In Section 3, we give a fixed-parameter algorithm for b-EDS on trees with the number of leaves as a parameter. This result implies that when the number of leaves is constant, then the problem can be solved in polynomial time. Therefore, if we want to prove that the problem is **NP**-hard, then a possible reduction should make use of trees with unbounded number of leaves.

To give a fixed-parameter algorithm for trees, the following fact plays an important role. On paths, there exists no gap between a natural integer programming formulation of b-EDS and its linear-programming relaxation. By using this fact, we will prove that b-EDS on trees with constant number of leaves can be formulated as a mixed integer program with constant number of integer variables. Thus, a fixed-parameter algorithm follows from the result of Lenstra [8] on fixed-parameter tractability of mixed integer programming when the number of integer variables is a parameter.

1.1.3 FPTAS for Trees

In Section 4, we give a fully polynomial-time approximation scheme (FPTAS, for short) for b-EDS on trees. Prior to our work, the best known approximation factor was two [1]. Thus, our algorithm improves the approximation factor drastically. This result implies that b-EDS on trees is not strongly **NP**-hard unless **P** = **NP**. Furthermore, if we want to prove the problem is **NP**-hard, then a possible proof cannot be done via a gap-preserving reduction from any **APX**-hard problem unless **P** = **NP**.

To give an FPTAS for trees, we generalize the problem. In this generalization, (i) there may exist parallel edges, (ii) each edge can be chosen at most once, and (iii) each edge has an "influence." If we choose an edge e with influence $p(e)$, then each edge in $\delta(e)$ is covered $p(e)$ times by e. We first give a pseudo-polynomial-time algorithm for this generalized problem on multitrees. Then, we give a reduction from b-EDS on trees to this generalized problem on multitrees. Finally, we prove that a pseudo-polynomial-time algorithm for this generalized problem on multitrees yields an FPTAS for b-EDS on trees.

1.2 Preliminaries: Polynomial-Time Algorithms and NP-Hardness

For the problem with numerical inputs, several notions of polynomial-time algorithms appear in the literature. Here, we summarize them.

Consider a problem in which we are given a combinatorial object O (in our case, a tree T) and a set of numbers (in our case $b(e)$ and $c(e)$ for all $e \in ET$) at most M. Suppose that the object O has size n (in our case T has n vertices).

An algorithm runs in *strongly polynomial time* if the running time is bounded by a polynomial in n. It runs in *weakly polynomial time* if the running time is bounded by a polynomial in n and $\log M$. It runs in *pseudo-polynomial time* if the running time is bounded by a polynomial in n and M. It is easy to see that a strongly polynomial-time algorithm is a weakly polynomial-time algorithm, and a weakly polynomial-time algorithm is a pseudo-polynomial-time algorithm. In this context, a *polynomial-time algorithm* is a weakly polynomial-time algorithm.

The problem is *strongly* **NP**-*hard* if it is **NP**-hard even if M is bounded by a polynomial in n. The usual **NP**-hardness is referred to as *weakly* **NP**-*hardness*.

2. Combinatorial Algorithm for Paths

In this section, we present a combinatorial strongly polynomial-time algorithm for b-EDS on a path P. Let $VP = \{v_1, v_2, \ldots, v_n\}$, $EP = \{e_1, e_2, \ldots, e_{n-1}\}$, and $e_i = (v_i, v_{i+1})$ for each $i \in \{1, 2, \ldots, n-1\}$. For each $i \in \{1, 2, \ldots, n\}$, we denote by P_i the subpath of P induced by $V_i = \{v_1, v_2, \ldots, v_i\}$. Let $B = \max\{b(e) \mid e \in EP\}$.

In the sequel, we first give a pseudo-polynomial-time algorithm for b-EDS on P (Sect. 2.1), and the time complexity will be improved to polynomial (Sect. 2.2).

2.1 Dynamic Programming

For each $i \in \{2, 3, \ldots, n\}$, each $x \in \{0, 1, \ldots, B\}$, and each $y \in \{0, 1, \ldots, 2B\}$, define $\rho_i(x, y)$ as the minimum cost of a vector s in $\mathbb{Z}_+^{EP_i}$ satisfying (i) $s(\delta(e_j)) \geq b(e_j)$ for every $j \in \{1, 2, \ldots, i-2\}$, (ii) $s(e_{i-1}) = x$, and (iii) $s(e_{i-2}) + y \geq b(e_{i-1})$, where $s(e_0) = 0$. If such a vector s does not exist, then define $\rho_i(x, y) = +\infty$. Roughly, these conditions mean that s satisfies the constraints for $e_1, e_2, \ldots, e_{i-1}$ if e_{i-1} is "chosen" x times and it is assumed to be "covered" $s(e_{i-2}) + y$ times.

We can compute $\rho_i(x, y)$ by using the following formulae. If $i = 2$, then (i) $\rho_2(x, y) = c(e_1) \cdot x$ if $y \geq b(e_1)$, and (ii) $\rho_2(x, y) = +\infty$ otherwise. If $i \in \{3, 4, \ldots, n\}$, then

$$\rho_i(x, y) = c(e_{i-1}) \cdot x + \min\{\rho_{i-1}(\hat{x}, \hat{y}) \mid \hat{y} - \hat{x} = x,\ \hat{x} \geq b(e_{i-1}) - y\}.$$

Then, the value $\rho_i(x, y)$ can be computed by looking at $O(B)$ val-

ues of ρ_{i-1} for each i, x, and y. Since we have $O(nB^2)$ choices of i, x, and y, we can compute all values of $\rho_i(x,y)$ in $O(nB^3)$ time. Since the optimal objective value of b-EDS on P is equal to $\min\{\rho_n(x,y) \mid x = y\}$, we can solve the problem in $O(nB^3)$ time, which is pseudo-polynomial time.

2.2 Compression of the DP Table

Since the DP table $\rho_i(x,y)$ has $\Theta(B^2)$ entries for each i, explicit computation of the table does not yield a polynomial-time algorithm. To achieve polynomiality, we construct a compact representation of ρ_i. More precisely, we represent ρ_i as a piecewise linear function such that the number of regions is bounded by a polynomial in n.

For each function $f\colon \mathbb{R}^2 \to \mathbb{R}\cup\{+\infty\}$, define $\operatorname{dom} f = \{(x,y) \in \mathbb{R}^2 \mid f(x,y) < +\infty\}$. A function $f\colon \mathbb{R}^2 \to \mathbb{R}\cup\{+\infty\}$ is said to be *convex* if it is convex in $\operatorname{dom} f$. A set $\mathcal{R} = \{R_1, R_2, \ldots, R_\ell\}$ of polygons (which might be unbounded) in $\mathbb{R}^2$ is called a *partition* of $\operatorname{dom} f$ if (i) $R_1 \cup R_2 \cup \cdots \cup R_\ell = \operatorname{dom} f$, and (ii) R_i and R_j do not intersect except at their boundaries for every distinct $i, j \in \{1,2,\ldots,\ell\}$. For each $i \in \{1,2,\ldots,\ell\}$ and each point (x,y) in R_i, suppose that a function $f\colon \mathbb{R}^2 \to \mathbb{R}\cup\{+\infty\}$ satisfies the constraint $f(x,y) = g_i(x,y)$, where $\mathcal{R} = \{R_1, R_2, \ldots, R_\ell\}$ is a partition of $\operatorname{dom} f$ and $g_i(x,y)$ is a linear function on $\mathbb{R}^2$. In such a case, f is said to be a *piecewise linear function with respect to* $\mathcal{R}$.

Our algorithm is based on the following lemmas. Recall that the domain of ρ_i is $\{0,1,\ldots,B\}\times\{0,1,\ldots,2B\}$. We now extend the domain of this function to $\mathbb{R}^2$. We define $\bar\rho_2\colon \mathbb{R}^2 \to \mathbb{R}\cup\{+\infty\}$ by (i) $\bar\rho_2(x,y) = c(e_1)\cdot x$ if $y \ge b(e_1)$ and $x \ge 0$, and (ii) $\bar\rho_2(x,y) = +\infty$ otherwise. For each $i \in \{3,4,\ldots,n\}$, define $\bar\rho_i\colon \mathbb{R}^2 \to \mathbb{R}\cup\{+\infty\}$ by

$$\bar\rho_i(x,y) = c(e_{i-1})\cdot x + \min\{\bar\rho_{i-1}(\hat x,\hat y) \mid \hat y - \hat x = x,\ \hat x \ge b(e_{i-1}) - y\}$$

if $x \ge 0$, and $\bar\rho_i(x,y) = +\infty$ otherwise. The following lemmas reveal fundamental properties of $\bar\rho_i$

Lemma 2.1. 1. *For every $i \in \{3,4,\ldots,n\}$, we have* $\operatorname{dom}\bar\rho_i = \{(x,y) \mid x \ge 0\}$.

2. *For every $i \in \{2,3,\ldots,n\}$, there exists a partition $\mathcal{R}_i$ of* $\operatorname{dom}\bar\rho_i$ *such that $\bar\rho_i$ is a convex piecewise linear function with respect to $\mathcal{R}_i$ and $|\mathcal{R}_i|$ is at most $(3/2)\cdot i(i-1)$. Such a partition $\mathcal{R}_i$ and a linear function on each part of $\mathcal{R}_i$ can be computed in strongly polynomial time.*

Lemma 2.2. *For every $i \in \{2,3,\ldots,n\}$, we have the following.*

1. *For every $x \ge 0$ and every $y_1, y_2 \ge B$, we have $\bar\rho_i(x,y_1) = \bar\rho_i(x,y_2)$.*
2. *For every $x \ge B$ and every $y \in \mathbb{R}$, we have $\bar\rho_i(x,y) \ge \bar\rho_i(B,y)$.*
3. *For every $x \in \{0,1,\ldots,B\}$ and every $y \in \{0,1,\ldots,2B\}$, we have $\bar\rho_i(x,y) = \rho_i(x,y)$.*

Assuming the lemmas, we may proceed as follows. By Lemma 2.1, we can compute in polynomial time $\mathcal{R}_i$ and a linear function on each part R in $\mathcal{R}_i$. Since $\bar\rho_i(x,y) = \rho_i(x,y)$ for every $x \in \{0,1,\ldots,B\}$ and every $y \in \{0,1,\ldots,2B\}$ by Lemma 2.2, we can compute the optimal value of b-EDS on P, which is $\min\{\rho_n(x,y) \mid x = y\}$, in strongly polynomial time.

Theorem 2.1. *We can solve b-*EDS *on paths in strongly polynomial time.*

We are left with proofs of Lemmas 2.1 and 2.2. Due to the limitation of the space, we omit the full proofs. We sketch basic ideas for the proofs below.

2.3 Good Partitions and Zigzags

In the proof of Lemma 2.1, we stick to a partition of a special kind. Namely, a partition $\mathcal{R}$ of $\operatorname{dom} f$ is called *good* if it satisfies the following.

> For each part R in $\mathcal{R}$, there exist $x_{\mathrm{d}}, y_{\mathrm{d}}, d_{\mathrm{d}}$ in $\mathbb{Z}\cup\{-\infty\}$ and $x^{\mathrm{u}}, y^{\mathrm{u}}, d^{\mathrm{u}}$ in $\mathbb{Z}\cup\{+\infty\}$ such that R is a polygon defined by $x_{\mathrm{d}} \le x \le x^{\mathrm{u}}$, $y_{\mathrm{d}} \le y \le y^{\mathrm{u}}$, and $d_{\mathrm{d}} \le y - x \le d^{\mathrm{u}}$.

The proof goes by induction on i. For fixed $i \ge 4$, assume that $\operatorname{dom}\bar\rho_{i-1} = \{(x,y) \mid x \ge 0\}$ and a good partition $\mathcal{R}_{i-1}$ as in the statement has been obtained. By the definition of $\bar\rho_i$, for every point $(x,y) \in \mathbb{R}^2$ with $x \ge 0$, we have

$$\begin{aligned}
\bar\rho_i(x,y) &= c(e_{i-1})\cdot x + \min\{\bar\rho_{i-1}(\hat x,\hat y) \mid \hat y - \hat x = x,\ \hat x \ge b(e_{i-1}) - y\}\\
&= c(e_{i-1})\cdot x +\\
&\quad \min_{\tilde y \le y} \min\{\bar\rho_{i-1}(\hat x,\hat y) \mid \hat y - \hat x = x,\ \hat x = b(e_{i-1}) - \tilde y\}\\
&= c(e_{i-1})\cdot x +\\
&\quad \min_{\tilde y \le y} \{\bar\rho_{i-1}(b(e_{i-1}) - \tilde y, b(e_{i-1}) - \tilde y + x)\}.
\end{aligned}$$

We consider the functions $g_1, g_2, g_3, g_4\colon \mathbb{R}^2 \to \mathbb{R}\cup\{+\infty\}$ defined by

$$\begin{aligned}
g_1(x,y) &= \bar\rho_{i-1}(b(e_{i-1}) - y, b(e_{i-1}) - y + x),\\
g_2(x,y) &= \min_{\tilde y \le y} g_1(x,\tilde y),\\
g_3(x,y) &= c(e_{i-1})\cdot x + g_2(x,y),\\
g_4(x,y) &= \begin{cases} g_3(x,y) & \text{if } x \ge 0\\ +\infty & \text{otherwise}\end{cases}
\end{aligned}$$

for each point (x,y) in $\mathbb{R}^2$. Then, we have $\bar\rho_i = g_4$.

We take a careful look at the influence of g_1, g_2, g_3, g_4 to the partition. The form of g_1 suggests a linear transformation of the coordinate system, and indeed, by the definition of a good partition, such a transformation maps a good partition of $\operatorname{dom}\rho_{i-1}$ to a good partition of $\operatorname{dom} g_1$. Assuming the existence of a good partition for $\operatorname{dom} g_2$, we can see that such a good partition is also a good partition of $\operatorname{dom} g_3$, from which we can easily obtain a good partition of $\operatorname{dom} g_4$. The size of partitions will not increase for those cases. Thus, the only problem may arise in finding a good partition of $\operatorname{dom} g_2$ from that of $\operatorname{dom} g_1$. This case needs a special treatment, and gives rise to a concept of zigzags.

Let $\mathcal{R} = \{R_1, R_2, \ldots, R_\ell\}$ be a partition of $\operatorname{dom} f$. The *boundary* of $\mathcal{R}$ is the union of the boundaries of $R_1, R_2, \ldots, R_\ell$. A *corner* of $\mathcal{R}$ is a point in $\mathbb{R}^2$ which is on the boundary of at least three regions in $\mathcal{R}\cup\{\mathbb{R}^2 \setminus \operatorname{dom} f\}$.

For a good partition $\mathcal{R}$ of $\operatorname{dom} f$, an *elementary zigzag through* $\mathcal{R}$ is a sequence $l_1, l_2, \ldots, l_t$ of line segments (l_1 and l_t may be rays) in $\operatorname{dom} f$ such that $l_1, l_2, \ldots, l_t$ are oriented so that the head of l_i and the tail of l_{i+1} coincide for each $i \in \{1,2,\ldots,t-1\}$, and they satisfy one of the following conditions (see Fig. 1).

a. The directions of l_i are $(1,1)$ and $(1,0)$, alternately. If the direction of l_i is $(1,0)$, then it is on the boundary of $\mathcal{R}$.
b. The directions of l_i are $(0,-1)$ and $(-1,-1)$, alternately. If

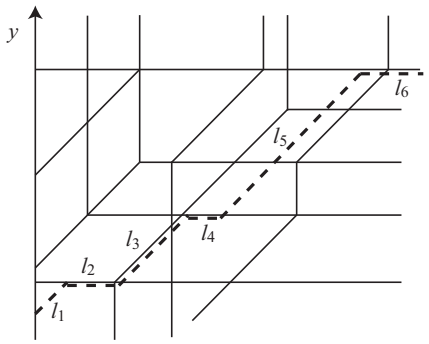

Fig. 1 A good partition and an elementary zigzag of type (a).

the direction of l_i is $(-1,-1)$, then it is on the boundary of $\mathcal{R}$.

c. The directions of l_i are $(-1,0)$ and $(0,1)$, alternately. If the direction of l_i is $(0,1)$, then it is on the boundary of $\mathcal{R}$.

Note that in the definition of elementary zigzags of type (a), l_i is not necessarily on the boundary of $\mathcal{R}$ if its direction is $(1,1)$. We also note that, elementary zigzags of types (b) and (c) are obtained from one of type (a) by linear transformation.

Our motivation for introducing elementary zigzags is the following lemma. We omit the proof.

Lemma 2.3. *Let $f\colon \mathbb{R}^2 \to \mathbb{R} \cup \{+\infty\}$ be a convex piecewise linear function with respect to a good partition $\mathcal{R}$ of* dom f. *Assume that, for any $x \in \mathbb{R}$, there exists a maximum minimizer $h(x)$ (i.e., a maximum of the minimizers) of a 1-dimensional function $f(x,\cdot)$. Then, $\{(x,h(x)) \mid x \in \mathbb{R}\}$ forms an elementary zigzag of type (a) on the boundary of $\mathcal{R}$.*

The construction of g_2 from g_1 can be done as follows. For fixed $x \in \mathbb{R}$, regard $g_1(x,\cdot)$ as a 1-dimensional function, and define $h(x)$ as the maximum minimizer of $g_1(x,\cdot)$. We may see that g_1 satisfies the condition in Lemma 2.3. Thus, $\{(x,h(x)) \mid x \in \mathbb{R}\}$ forms an elementary zigzag Z of type (a) on the boundary of a good partition of $\mathrm{dom}\, g_1$. In particular, $g_2(x,y) = g_1(x,y)$ if $y \le h(x)$ and $g_2(x,y) = g_1(x,h(x))$ if $y > h(x)$. Therefore, the size of Z will contribute to the size of a good partition of $\mathrm{dom}\, g_2$. This also means that we need to keep track of the size of such an elementary zigzag in the induction. To this end, we actually prove a stronger statement than Lemma 2.1, and use a more generalized concept.

3. Fixed-Parameter Algorithm for Trees

In this section, we give a fixed-parameter algorithm for b-EDS on trees with the number of leaves as a parameter. We first consider a subproblem on a path, which can be solved by linear programming, and then describe how to formulate b-EDS on trees using these subproblems.

3.1 Subproblem

Let P be a path such that $VP = \{v_1, v_2, \ldots, v_n\}$, $EP = \{e_1, e_2, \ldots, e_{n-1}\}$, and $e_i = (v_i, v_{i+1})$ for each $i \in \{1,2,\ldots,n-1\}$. Given integers $y(e_1), y(e_{n-1}), z(v_1)$, and $z(v_n)$, we consider the problem of finding x in $\mathbb{Z}_+^{EP}$ that minimizes $\langle c, x\rangle$ subject to the following: when $n \ge 3$,

$$\begin{aligned} x(e_{i-1}) + x(e_i) + x(e_{i+1}) &\ge b(e_i) \quad (\forall i \in \{2,3,\ldots,n-2\}),\\ x(e_2) + z(v_1) \ge b(e_1), &\quad x(e_{n-2}) + z(v_n) \ge b(e_{n-1}),\\ x(e_1) = y(e_1), &\quad x(e_{n-1}) = y(e_{n-1}), \end{aligned} \tag{1}$$

and, when $n = 2$,

$$x(e_1) \le z(v_1) + z(v_2) - b(e_1), \quad x(e_1) = y(e_1). \tag{2}$$

Note that these constraints mean that $x(e_1) = y(e_1)$, $x(e_{n-1}) = y(e_{n-1})$, and if v_1 and v_n are "covered" at least $z(v_1)$ and $z(v_n)$ times, respectively, then x is a feasible solution of b-EDS on P. This problem can be written as the following integer program:

$$\min\{\langle c, x\rangle \mid Ax \ge b',\ x \in \mathbb{Z}_+^{EP}\}, \tag{3}$$

where A is a 0-1 matrix and each entry of b' is integer if $y(e_1), y(e_{n-1}), z(v_1)$, and $z(v_n)$ are integers. Since A is totally unimodular (A is an interval matrix) [10], the optimal value of the problem is equal to that of its linear-programming relaxation.

3.2 Algorithm

Let T be a given tree with at most ℓ leaves. Let L be the set of all leaves, and let S be the set of all *hub vertices* of VT (i.e., vertices of degree at least three). Define $F := \bigcup\{\delta(v) \mid v \in L \cup S\}$. We denote by $\mathcal{P}$ the set of all paths P such that both end vertices of P are in $L \cup S$ and no inner vertex of P is in $L \cup S$. It is not difficult to see that $\mathcal{P}$ gives a partition of ET. For each path P in $\mathcal{P}$, we denote by b_P the restriction of b to EP. We note that $|L| \le \ell$, $|S| \le \ell$, $|\mathcal{P}| \le 2\ell$, and $|F| \le 2|\mathcal{P}| \le 4\ell$.

It is not difficult to see that b-EDS is equivalent to the problem of finding vectors x in $\mathbb{Z}_+^{ET}$, y in $\mathbb{Z}_+^{F}$, and z in $\mathbb{Z}_+^{L\cup S}$ that minimize $\langle c, x\rangle$ subject to

$$\begin{aligned} &z(v) = y(\delta(v)) \text{ for each vertex } v \text{ in } L \cup S,\\ &\text{Condition (1) for each path } P \text{ in } \mathcal{P} \text{ of length at least two,}\\ &\text{Condition (2) for each path } P \text{ in } \mathcal{P} \text{ of length one.} \end{aligned} \tag{4}$$

By the arguments in the previous subsection, the optimal value does not change even if we drop the integrality of x. That is, b-EDS on T is equivalent to minimizing $\langle c, x\rangle$ subject to $x \in \mathbb{R}_+^{ET}$, $y \in \mathbb{Z}_+^{F}$, $z \in \mathbb{Z}_+^{L\cup S}$, and (4) , which is a mixed integer program. Indeed, suppose we have an optimal solution x in $\mathbb{R}_+^{ET}$, y in $\mathbb{Z}_+^{F}$, and z in $\mathbb{Z}_+^{L\cup S}$ for this mixed integer program. Then, for the fixed y in $\mathbb{Z}_+^{F}$ and z in $\mathbb{Z}_+^{L\cup S}$, the restriction x_P of x on each path P in $\mathcal{P}$ is an optimal solution to the linear-programming relaxation of (3) , and hence there exists an integer optimal solution to the linear-programming relaxation because the coefficient matrix is totally unimodular. Since the number of integer variables in this mixed linear program is bounded by $|F| + |L \cup S| \le 6\ell$, it can be solved in polynomial time by the following theorem (see also [10]).

Theorem 3.1 (Lenstra [8])**.** *The problem of solving* $\max\{\langle d_1, x\rangle + \langle d_2, y\rangle \mid A_1x + A_2y \le b',\ x \text{ is integral}\}$, *where A_1 has rank at most ℓ, is fixed-parameter tractable with respect to ℓ.*

Note that the current best running time is $2^{O(\ell \log \ell)}$ times a polynomial of the input size [4], [5]. By using this theorem, we can obtain the following result.

Theorem 3.2. *The problem b-EDS is fixed-parameter tractable with respect to the number of leaves of an input tree.*

4. FPTAS for Trees

We denote by $\mathsf{OPT}(T)$ the optimal objective value for T. Precisely speaking, $\mathsf{OPT}(T)$ depends on b and c. However, we omit them for brevity. To give an FPTAS, we first introduce a generalization of b-EDS (b-GEDS, for short), and construct a pseudo-polynomial-time algorithm to solve b-GEDS. Based on the algorithm, we then give an FPTAS for b-EDS on trees.

In b-GEDS, an input graph G may have parallel edges. In addition to $b(e)$ and $c(e)$, each edge e is associated with a nonnegative integer $p(e)$, called the *influence* of e. A subset D of E is called a *generalized b-edge dominating set* of G, if $p(\delta(e) \cap D) \geq b(e)$ for every edge e of G. The *cost* of a generalized b-edge dominating set D of G, denoted by $\mathsf{cost}(D)$, is defined as $c(D)$. The goal of b-GEDS is to find a generalized b-edge dominating set with the minimum cost; we denote by $\mathsf{GOPT}(G)$ the optimal objective value for a given multigraph G. We can prove that b-GEDS is weakly **NP**-hard on multitrees consisting of two vertices (we omit the proof). Notice that a graph T is called a *multitree* if it is a tree when we regard parallel edges as a single edge. We will show in Sect. 4.2 that b-EDS on trees can be reduced to b-GEDS on multitrees.

We note here that a pseudo-polynomial-time algorithm for b-EDS on trees can be obtained without resorting to b-GEDS. However, such an algorithm is not suited for applying the scale-and-round technique [7], [12] to obtain an FPTAS since bounding the error seems difficult; a similar situation occurs in the bounded knapsack problem [7], Chapter 7.

4.1 Pseudo-Polynomial-Time Algorithm for b-GEDS on Multitrees

In this subsection, we prove the following theorem.

Theorem 4.1. *On a multitree T with m edges, b-GEDS can be solved in $O(mB^4)$ time, where $B := \max\{b(e) \mid e \in ET\}$.*

Proof. We give an algorithm to solve b-GEDS in the desired time complexity. Our algorithm simply computes $\mathsf{GOPT}(T)$. It is easy to modify our algorithm so that it finds a generalized b-edge dominating set with the minimum cost $\mathsf{GOPT}(T)$.

We choose an arbitrary vertex r as the root of T, and regard T as a rooted tree. For each vertex v of T, let T_v be the subtree of T which is rooted at v and is induced by v and all descendants of v on T. Let $w_1, w_2, \ldots, w_q$ be the children of v, ordered arbitrarily. For each $j \in \{1, 2, \ldots, q\}$, we denote by T_v^j the subtree of T induced by $\{v\} \cup VT_{w_1} \cup VT_{w_2} \cup \cdots \cup VT_{w_j}$. For the sake of notational convenience, we denote by T_v^0 the tree consisting of a single vertex v. Then, $T_v = T_v^0$ for each leaf v of T. Let E_{vw_j} be the set of (multiple) edges joining v and w_j.

For each vertex v of T, each $j \in \{0, 1, \ldots, q\}$, and each $x, y \in \{0, 1, \ldots, B\}$, we define $h_v^j(x, y)$ as the minimum cost of a subset D_v^j of ET_v^j subject to

1. $p(\delta(e) \cap D_v^j) \geq b(e)$ $\quad (\forall e \in ET_{w_k},\ \forall k \in \{1, 2, \ldots, j\})$,
2. $p(\delta(v) \cap D_v^j) \geq x$, and
3. $y + p(\delta(e) \cap ET_{w_k} \cap D_v^j) \geq b(e)$ $\quad (\forall e \in E_{vw_k},\ \forall k \in \{1, 2, \ldots, j\})$.

If such a subset D_v^j does not exist, then define $h_v^j(x, y) = +\infty$. Roughly speaking, x guarantees the influence on v from $\delta(v) \cap ET_v^j$, and y specifies the total influence guaranteed around v in T. For the notational convenience, we sometimes denote $h_v(x, y) = h_v^q(x, y)$, where q is the number of children of v.

The proposed algorithm computes $h_v^j(x, y)$ for each vertex v of T, each $j \in \{0, 1, \ldots, q\}$, and all $x, y \in \{0, 1, \ldots, B\}$, from the leaves of T to the root r of T, by means of dynamic programming. Since $T_r = T$ for the root r of T, we can compute $\mathsf{GOPT}(T) = \min\{h_r(x, y) \mid x = y \in \{0, 1, \ldots, B\}\}$ for a given multitree T. Therefore, $\mathsf{GOPT}(T)$ can be computed in $O(B)$ time if we have computed the values $h_r(x, y)$ for all $x, y \in \{0, 1, \ldots, B\}$.

We now explain how to compute $h_v^j(x, y)$. We first consider the case where $j = 0$. Recall that, for each vertex v of T, the subtree T_v^0 is defined to be a single vertex v. As the initialization, for each $x, y \in \{0, 1, \ldots, B\}$, define $h_v^0(x, y) = 0$. This can be done in $O(nB^2)$ time for all vertices of T and all $x, y \in \{0, 1, \ldots, B\}$.

We then consider the case where $j \geq 1$, and hence v is an internal vertex of T. Suppose that we have already computed $h_v^{j-1}(x, y)$ and $h_{w_j}(x, y)$ for all $x, y \in \{0, 1, \ldots, B\}$. For each $z \in \{0, 1, \ldots, B\}$, let

$$\pi(E_{vw_j}, z) = \min\{c(D) \mid D \subseteq E_{vw_j},\ p(D) \geq z\}.$$

If such a subset D does not exist, then we define $\pi(E_{vw_j}, z) = +\infty$. By a simple dynamic-programming algorithm similar to the knapsack problem [7], we can compute $\pi(E_{vw_j}, z)$ for all $z \in \{0, 1, \ldots, B\}$ in $O(|E_{vw_j}|B)$ time.

We now compute $h_v^j(x, y)$ for each $x, y \in \{0, 1, \ldots, B\}$. We first fix the influence $z \in \{0, 1, \ldots, x\}$ obtained by the choice of E_{vw_j}, and compute the following value $h_v^j(x, y; z)$:

$$h_v^j(x, y; z) = \pi(E_{vw_j}, z) + \min\{h_v^{j-1}(x - z, y) + h_{w_j}(x', x' + z)\}, \quad (5)$$

where the minimum above is taken over all $x' \in \{0, 1, \ldots, B\}$ such that $y + x' \geq \max\{b(e) \mid e \in E_{vw_j}\}$; let $h_v^j(x, y; z) = +\infty$ if such an integer x' does not exist. Then, we have

$$h_v^j(x, y) = \min\{h_v^j(x, y; z) \mid z \in \{0, 1, \ldots, x\}\}. \quad (6)$$

We estimate the running time of this update computation. Recall that we can compute the values $\pi(E_{vw_j}, z)$ for all $z \in \{0, 1, \ldots, B\}$ in $O(|E_{vw_j}|B)$ time. For each $x, y \in \{0, 1, \ldots, B\}$, by (5) and (6), $h_v^j(x, y)$ can be computed in $O(B^2)$ time. Thus, for each vertex v of T and each $j \in \{1, 2, \ldots, q\}$, we can compute $h_v^j(x, y)$ for all $x, y \in \{0, 1, \ldots, B\}$ in $O(|E_{vw_j}|B + B^4)$ time. Then, $h_r(x, y)$ for the root r can be computed in total time

$$\sum_{v \in VT} \sum_{j} O(|E_{vw_j}|B + B^4) = O(mB + nB^4) = O(mB^4),$$

where $n = |VT|$ and $m = |ET|$. Notice that $n \leq m + 1$ since T is a multitree. This completes the proof of Theorem 4.1. □ □

4.2 FPTAS for b-EDS on Trees

By using the following proposition, we can reduce b-EDS to b-GEDS. We omit its proof.

Proposition 4.1. *Let (G', b', c') be an instance of b'-EDS with $B' = \max\{b'(e) \mid e \in EG'\}$. Then, one can construct an instance (G, b, c, p) of b-GEDS in polynomial time such that*

a. *$|VG| = |VG'|$ and $|EG| = |EG'| \cdot (\lfloor \log B' \rfloor + 1)$,*

b. $\max\{b(e) \mid e \in EG\} = B'$, *and*
c. *there exists a* b'*-edge dominating vector* s *of* G' *with* $\mathsf{cost}(s) = \gamma$ *if and only if there exists a generalized* b*-edge dominating set* D *of* G *with* $\mathsf{cost}(D) = \gamma$.

In particular, $\mathsf{GOPT}(G) = \mathsf{OPT}(G')$ *holds.*

To obtain an FPTAS for b-EDS on trees, we now rephrase Theorem 4.1 when restricted to the instances corresponding to those of b-EDS. We omit the proof.

Lemma 4.1. *Let* T *be the multitree obtained by Proposition* 4.1 *from a tree* T' *of an instance of* b'-EDS. *Let* U *be any upper bound on* $\mathsf{GOPT}(T)$. *Then,* b-GEDS *on* T *can be solved in* $O(mU^4)$ *time, where* $m = |ET|$.

We finally give an FPTAS for b-EDS on trees, based on the pseudo-polynomial-time algorithm in Sect. 4.1.

Theorem 4.2. *There exists an FPTAS for* b-EDS *on trees.*

Our idea for the FPTAS is to combine techniques developed for the bounded knapsack problem [7], Chapter 7 and the scale-and-round technique [7], [12]. Due to the limitation of the space, we omit the detail.

Acknowledgments The authors would like to thank Yusuke Matsumoto and Chien-Chung Huang for helpful discussions on this topic. The first author is supported by JSPS KAKENHI Grant Numbers 25106504, 25330003. The second author is supported by JST, ERATO, Kawarabayashi Large Graph Project, and by JSPS KAKENHI Grant Numbers 25730001, 24106002. The third author is supported by JSPS KAKENHI Grant Number 24106005. The fourth author is supported by JST, ERATO, Kawarabayashi Large Graph Project, and by JSPS KAKENHI Grant Numbers 24106002, 24700004. The fifth author is supported by Grant-in-Aid for Scientific Research from Ministry of Education, Science and Culture, Japan, and Japan Society for the Promotion of Science (JSPS).

References

[1] Berger, A., Fukunaga, T., Nagamochi, H., Parekh, O.: Approximability of the capacitated-edge dominating set problem. Theor. Comput. Sci. 385(1-3), 202–213 (2007)
[2] Berger, A., Parekh, O.: Linear time algorithms for generalized edge dominating set problems. Algorithmica 50(2), 244–254 (2008)
[3] Berger, A., Parekh, O.: Erratum to: Linear Time Algorithms for Generalized Edge Dominating Set Problems. Algorithmica 62(1), 633–634 (2012)
[4] Dadush, D., Peikert, C., Vempala, S.: Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In FOCS, 580–589 (2011)
[5] Dadush, D., Vempala, S.: Deterministic construction of an approximate M-ellipsoid and its applications to derandomizing lattice algorithms. In SODA, 1445–1456 (2012)
[6] Fujito, T., Nagamochi, H.: A 2-approximation algorithm for the minimum weight edge dominating set problem. Discrete Appl. Math. 118(3), 199–207 (2002)
[7] Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer-Verlag (2004)
[8] Lenstra, H. W.: Integer programming with a fixed number of variables. Math. Oper. Res. 8(4), 538–548 (1983)
[9] Parekh, O.: Edge dominating and hypomatchable sets. In SODA, 287–291 (2002)
[10] Schrijver, A.: Theory of Linear and Integer Programming. John Wiley & Sons (1986)
[11] Tardos, É.: A strongly polynomial algorithm to solve combinatorial linear programs. Operations Research. 34(2), 250–256 (1986)
[12] Vazirani, V.V.: Approximation algorithms. Springer-Verlag (2001)
[13] Yannakakis, M., Gavril, F.: Edge dominating sets in graphs. SIAM J. Appl. Math. 38(3), 364–372 (1980)