Improved Bounds for Online Dominating Sets of Trees[*][†]

Koji M. Kobayashi

National Institute of Informatics, Tokyo, Japan
kobaya@nii.ac.jp

Abstract

The online dominating set problem is an online variant of the minimum dominating set problem, which is one of the most important NP-hard problems on graphs. This problem is defined as follows: Given an undirected graph $G = (V, E)$, in which V is a set of vertices and E is a set of edges. We say that a set $D \subseteq V$ of vertices is a *dominating set* of G if for each $v \in V \setminus D$, there exists a vertex $u \in D$ such that $\{u, v\} \in E$. The vertices are revealed to an online algorithm one by one over time. When a vertex is revealed, edges between the vertex and vertices revealed in the past are also revealed. A revealed subtree is connected at any time. Immediately after the revelation of each vertex, an online algorithm can irrevocably choose vertices which were already revealed and must maintain a dominating set of a graph revealed so far. The cost of an algorithm on a given tree is the number of vertices chosen by it, and its objective is to minimize the cost. Eidenbenz (Technical report, Institute of Theoretical Computer Science, ETH Zürich, 2002) and Boyar et al. (SWAT 2016) studied the case in which given graphs are trees. They designed a deterministic online algorithm whose competitive ratio is at most three, and proved that a lower bound on the competitive ratio of any deterministic algorithm is two.

In this paper, we also focus on trees. We establish a matching lower bound for any deterministic algorithm. Moreover, we design a randomized online algorithm whose competitive ratio is exactly $5/2 = 2.5$, and show that the competitive ratio of any randomized algorithm is at least $4/3 \approx 1.333$.

1998 ACM Subject Classification F.1.2 Modes of Computation, Online Computation, G.2.2 Graph Theory, Graph Algorithms, I.1.2 Algorithms, Analysis of Algorithms

Keywords and phrases online algorithm, dominating set, competitive analysis, tree graph, randomized algorithm

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2017.52

1 Introduction

The dominating set problem is one of the most important NP-hard problems on graphs. This problem is defined as follows: Given an undirected graph $G = (V, E)$, in which V is a set of vertices and E is a set of edges. We say that a set $D \subseteq V$ of vertices is a *dominating set* of G if for each vertex $v \in V \setminus D$, there exists a vertex $u \in D$ such that $\{u, v\} \in E$. The objective of the problem is to construct a minimum dominating set. This problem has been extensively studied for many applications, such as communication in ad-hoc networks (see e.g., [16]) and facility location on networks (e.g., [12]).

The dominating set problem has also been studied in online settings [11, 4, 2]. In one of the settings [4, 2], vertices are revealed to an online algorithm one by one, and edges

[*] This work was supported by JSPS KAKENHI Grant Number 26730008.
[†] A full version of the paper is available at https://arxiv.org/abs/1710.11414.

between a revealed vertex and vertices revealed in the past are also revealed. The *input* of this setting is an undirected graph and a sequence consisting of all the vertices of the graph. (This sequence represents an order of the vertices revealed to an online algorithm.) An online algorithm holds the empty set U at the beginning. When a new vertex is revealed, the algorithm can add vertices revealed so far to U, which means that an added vertex is not necessarily the newly revealed one. The algorithm must not remove a vertex from U. The total number of vertices is not known to an online algorithm before the final vertex is revealed. Thus, U must be a dominating set immediately after the revelation of each vertex. The performance of online algorithms is evaluated using *competitive analysis* [1, 14]. The *cost* of an algorithm ALG for an input σ is the size of a dominating set constructed by ALG for σ, which is denoted as $C_{ALG}(\sigma)$. We say that the (strict) competitive ratio of an online algorithm ON is at most c or ON is *c-competitive* if for any input σ, $C_{ON}(\sigma) \leq cC_{OPT}(\sigma)$, in which OPT is an optimal offline algorithm for σ. If ON uses randomization, the expected cost of ON is used.

Previous Results and Our Results

For trees, Eidenbenz [4] and Boyar et al. [2] designed a 3-competitive deterministic algorithm, and proved that the competitive ratio of any deterministic online algorithm is at least two (Boyar et al. showed their results in terms of asymptotic competitive ratios, but the results can hold for strict competitive ratios as well).

In this paper, we show the following three results for trees: (i) We prove that a lower bound on the competitive ratio of any deterministic algorithm is three. This bound matches the above upper bound. (ii) We establish a randomized online algorithm whose competitive ratio is exactly $5/2 = 2.5$. This algorithm is the first non-trivial randomized algorithm for the online dominating set problem for any graph class. (iii) We show that the competitive ratio of any randomized algorithm is at least $4/3 \approx 1.333$. The above results are shown with respect to the *strict* competitive ratio. However, it is easy to see that the same results for the *asymptotic* competitive ratios as (i) and (iii) can be shown in a quite similar way to their proofs. (Note that any upper bound on the strict competitive ratio is an upper bound on that on the asymptotic competitive ratio. That is, (ii) holds for the asymptotic competitive ratio.)

Related Results

For several graph classes, Eidenbenz [4] and Boyar et al. [2] studied online algorithms of a few variants of dominating sets, namely, connected dominating sets, total dominating sets and independent dominating sets. Their results are summarized in the table in Sec. 6 of [4] and Table 2 in Sec. 1 of [2]. For example, they proved that the optimal competitive ratios on a bipartite graph and a planar graph are $n - 1$, in which n is the number of given vertices. Boyar et al. [2] defined an *incremental* algorithm as an algorithm which maintains a dominating set immediately after a new vertex is revealed. An online algorithm is incremental, but an optimal incremental algorithm knows the whole input and can perform better than any online algorithm. They measured the performance of online algorithms compared with an optimal *incremental* algorithm in addition to an optimal offline algorithm. Moreover, they compared the performance of an optimal incremental algorithm with that of an optimal offline algorithm for several graph classes, which is also summarized in Table 1 in Sec. 1 of [2].

King and Tzeng [11] studied two different variants of online dominating sets on general graphs. One variant is the same as the one studied in this paper, except that immediately after a new vertex is revealed, an online algorithm can choose the new one but cannot choose vertices revealed previously. In this setting, they designed a deterministic algorithm whose competitive ratio is at most $n-1$, and proved that the algorithm is the best possible. In the other variant, an online algorithm knows all vertices in advance, and at a time i, all the edges between the i-th vertex v_i and the other vertices are revealed. They showed an upper bound of $3\sqrt{n}/2$ and a lower bound of of $\sqrt{n}$ for this variant.

For the offline setting, the minimum dominating set problem is one of the most significant NP-hard problems on graphs and has been widely studied. One of the most important open problems is to develop exact (exponential) algorithms (see, e.g. [5, 6, 8, 13, 15, 10]). The current fastest algorithm solves this problem in $O(1.4864^n)$ time and polynomial space [10]. Moreover, many variants have been proposed by putting additional constraints on the original dominating set problem and have been extensively studied: for example, connected domination, independent domination and total domination (see, .e.g. [3],[7] and [9], respectively).

2 Preliminaries

2.1 Model Description

We are given an undirected tree and its vertices are revealed to an online algorithm one by one over time. The total number of the vertices is not known to the online algorithm up to the end of the input. When the i-th vertex v_i is revealed to the online algorithm, all the edges between v_i and v_j such that $j < i$ are also revealed. Except for the first revealed vertex, a newly revealed vertex has exactly one edge to a vertex revealed previously. That is, a revealed subtree is connected at any time. An *input* of the problem is a three-tuple of the form (V, E, S), in which V is the set of all the vertices of a given tree, E is the set of all the undirected edges of the tree, and S is a sequence consisting of all the vertices in V. S represents an order of the vertices revealed to an online algorithm. An algorithm has the empty set U before the first vertex is revealed. The algorithm can add vertices into U immediately after the revelation of each vertex, and it is necessary for U to be a dominating set of the given tree at the end of the input. If the algorithm is online, it does not know when the input has ended, and thus U must be a dominating set immediately after each vertex is revealed. Once a vertex is added into U, it must not be removed from U later. The *cost* of the algorithm for an input σ is the number of vertices in U at the end of σ, and the objective of the problem is to minimize the cost. We evaluate the performance of an online algorithm using competitive analysis. We say that the *competitive ratio* of a deterministic online algorithm ON is at most c if for any input σ, $C_{ON}(\sigma) \leq cC_{OPT}(\sigma)$. If ON is a randomized online algorithm, then the expected cost of ON is used, which is denoted by $\mathbb{E}[C_{ON}(\sigma)]$. If for any input σ, $\mathbb{E}[C_{ON}(\sigma)] \leq cC_{OPT}(\sigma)$, then we say that the competitive ratio of a randomized online algorithm ON is at most c against any oblivious adversary.

If the number of vertices in a given tree is one, the cost ratio of any algorithm is clearly one. Thus, we assume that this number is at least two.

2.2 Notation and Definitions

In this section, we give some definitions and notation used throughout this paper. For any $i(= 1, 2, \ldots)$, we use v_i to denote the i-th revealed vertex to an online algorithm (the first revealed vertex v_1 appears frequently in this paper). We say that vertices v and u are *adjacent* if $\{v, u\} \in E$, in which E is the set of all the edges of a given graph. When a vertex v is revealed such that v is adjacent to a vertex u which was revealed before v, then we say that v *arrives* at u. For any vertex v and any online algorithm ON, $D_{ON}(v)$ denotes a dominating set constructed by ON of a revealed graph up to the time of the revelation of v. We will omit ON from the notation when it is clear from the context. For an algorithm ALG including an offline algorithm, $D_{ALG}(\sigma)$ denotes a dominating set constructed by ALG after the end of the input σ. We will omit σ from the notation when it is clear from the context. For a vertex v, we say that ALG *selects* v if $v \in D_{ALG}$. For vertices u and v such that u is revealed after v, $deg_u(v)$ denotes the degree of v immediately after u is revealed. $deg(v)$ denotes the degree of v after the end of the input. For a vertex v and a vertex u revealed after v, we say that u is a *descendant* of v if any vertex on the simple path from v to u is revealed after v. The *cost* of a deterministic algorithm ALG for a vertex set U is the number of vertices selected by ALG in U. That is, it is the number of vertices in $U \cap D_{ALG}$. Moreover, if U contains only one vertex, then we simply say the *cost for the vertex*. In the same way, we use the term "the expected cost of ALG for U (or a vertex)" if ALG is a randomized algorithm.

3 Deterministic Lower Bound

Due to page limitations, we omit most of the proofs of the following lemmas and theorems. The full version of this paper is available at `https://arxiv.org/abs/1710.11414`.

3.1 Overview of Proof

We first outline an input to obtain our lower bound. The tree of the input is constructed according to two routines. The tree can be divided into several subtrees satisfying some properties and we evaluate the competitive ratio for each set of subtrees. One of the routines appoints a vertex as the root to construct a subtree, which is called a *base vertex*. The other routine constructs several subtrees with at most two leaves, each of which arises from the base vertex. The set of all the vertices excluding the root in each of the subtrees is called a *T-set*. It depends on the behavior of an online algorithm ON how many T-sets are constructed and how many leaves and inner vertices composing T-sets are. If a T-set contains two leaves, the leaves share the adjacent vertex. For each T-set, OPT selects one vertex for every consecutive three vertices starting with the parent of a leaf in it. If the degree of a vertex selected by OPT is two, ON selects the vertex and the two adjacent vertices. Otherwise, that is its degree is at least three, ON selects at least three vertices from the vertex and all the adjacent vertices.

Let us explain the proof more in detail. If a T-set contains sufficiently many inner vertices, it is called a *T_3-set*. Otherwise, a T-set such that ℓ modulo $3 = i$ is called a *T_i-set*, in which ℓ is the length from the base vertex to a leaf in the T-set. One of the routines tries to force ON to construct one of the following four sets of T-sets from a base vertex (Fig. 1): (1) a set of two T_1-sets and at least zero T_0-set, (2) a set of one T_2-set and at least zero T_0-set, (3) a set of one T_3-set, at most one T_1-set and at least zero T_0-set, and (4) a set of sufficiently many T_0-sets and at most one T_1-set. The cost ratios of these T-sets are three for (1) or (2) and approximately three for (3) or (4), respectively. ON can construct none of these sets. Namely, (5) ON constructs one T_1-set and then does one T_2-set (Further, ON may also construct T_0-sets). In this case, the routine partitions the T_2-set into a vertex u, a T_1-set and a T_0-set. This T_0-set and all the T-sets in (5) except for the partitioned T_2-set compose a set of T-sets of (1). Then, the routine finishes constructing a subtree from the current

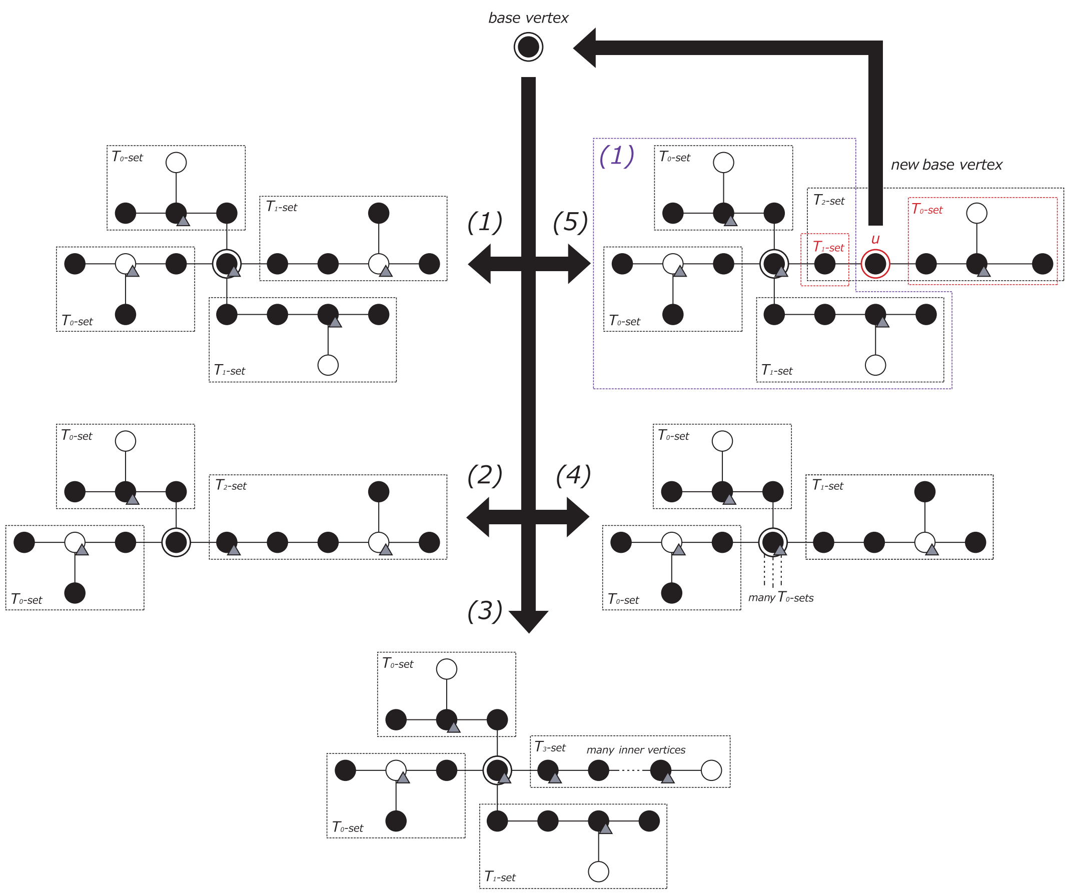

Figure 1 An example of the five sets of T-sets from (1) to (5). Highlighted vertices denote base vertices. Black vertices denote vertices selected by ON. Vertices with a gray triangle denote vertices selected by OPT. One of the five sets of T-sets is constructed for a base vertex. If (5) is constructed, the T_2-set in the set is partitioned into a new base vertex u, a T_1-set and a T_0-set. After that, the routines force ON to construct one of the five sets of T-sets for u recursively. u is not dominated by OPT yet, but is dominated later.

base vertex, whose cost ratio is three, and appoints u as a new base vertex. One T_0-set, which is constructed from the above partition of the T_2-set, belongs to the new base vertex u. Since the set of T-sets of u is not classified into any of the above four categories, the routine continues to construct subtrees for u. This is how the routine tries to construct one of the four sets of T-sets for all base vertices and to achieve a lower bound of (approximately) three. Therefore, we have the following theorem:

▶ **Theorem 1.** *For any $\varepsilon > 0$, the competitive ratio of any deterministic online algorithm is at least $3 - \varepsilon$.*

4 Randomized Upper Bound

4.1 Algorithm

First, we define our algorithm RA. Before the first vertex is revealed, RA chooses to start running one of two deterministic online algorithms A and B, which are defined later, with the probability of $1/2$ and thereafter keeps running it up to the end of the input. For a vertex v, $p(v)$ denotes the length of the simple path from v_1 to v. Roughly speaking, the difference between A and B is that for a vertex v, A selects v if $p(v)$ is odd, and B selects v if $p(v)$ is even. Then, A and B try to establish the property that for any vertex u of degree at most two, $u \notin D_A \cap D_B$.

A (B) can select a vertex which A (B) selected previously in the following definition. It means that A (B) does nothing at that time. First, we give the definition of A as follows.

Algorithm 1: Algorithm A

```
Suppose that the $i$-th vertex $v_i$ is revealed.
**Case 1 ($i = 1$):** Select $v_1$.
**Case 2 ($i \geq 2$):** Suppose that $v_i$ arrives at a vertex $u$.
  **Case 2.1 ($deg_{v_i}(u) \geq 3$):** Select $u$.
  **Case 2.2 ($deg_{v_i}(u) \leq 2$):**
    **Case 2.2.1 ($p(v_i)$ modulo $2 = 0$):** Select $u$.
    **Case 2.2.2 ($p(v_i)$ modulo $2 = 1$):** Select $v_i$.
```

Since A selects either a revealed vertex v_i or the vertex adjacent to v_i, the set of vertices selected by A is a dominating set of a revealed graph immediately after each of A's selections.

The definition of B is quite the same as that of A except for Case 2.2. The process of B in Case 2.2.1 (2.2.2) is the same as that of A in Case 2.2.2 (2.2.1). That is, B selects v_i and u in Cases 2.2.1 and 2.2.2, respectively. Thus, the set of vertices selected by B is also a dominating set at any time. We omit its formal definition due to page limitation.

4.2 Basic Properties of RA

In this section, we show several basic properties of dominating sets by A and B.

▶ **Lemma 2.** *The following properties hold for a vertex v:*

(1) *If $v = v_1$, $v \in D_A$ and $v \in D_B$.*

Suppose that $v \neq v_1$.

(2) *If $deg(v) \geq 3$, $v \in D_A$ and $v \in D_B$.*

(3) *Suppose that $deg(v) = 2$.*

(3-e) *If $p(v)$ modulo $2 = 0$, $v \notin D_A$ and $v \in D_B$.*

(3-o) *If $p(v)$ modulo $2 = 1$, $v \in D_A$ and $v \notin D_B$.*

(4) *Suppose that $deg(v) = 1$ and let $\tilde{u}$ be the vertex adjacent to v.*

(4-1) *Suppose that $deg(\tilde{u}) \geq 3$ and $deg_v(\tilde{u}) \leq 2$.*

(4-1-e) *If $p(v)$ modulo $2 = 0$, $v \notin D_A$ and $v \in D_B$.*

(4-1-o) *If $p(v)$ modulo $2 = 1$, $v \in D_A$ and $v \notin D_B$.*

(4-2) *If $deg(\tilde{u}) \geq 3$ and $deg_v(\tilde{u}) \geq 3$, then $v \notin D_A$ and $v \notin D_B$.*

(4-3) *Suppose that $deg(\tilde{u}) \leq 2$.*

(4-3-e) *If $p(v)$ modulo $2 = 0$, $v \notin D_A$ and $v \in D_B$.*

(4-3-o) *If $p(v)$ modulo $2 = 1$, $v \in D_A$ and $v \notin D_B$.*

▶ **Lemma 3.** *The expected cost of RA for v is as follows:*

(1) *If $v = v_1$, it is one.*

Suppose that $v \neq v_1$.

(2) *If $deg(v) \geq 3$, it is one.*

(3) *If $deg(v) = 2$, it is $1/2$.*

(4) *Suppose that $deg(v) = 1$ and v is adjacent to a vertex u.*

(4-1) *If $deg(u) \geq 3$ and $deg_v(u) \leq 2$, it is $1/2$.*
(4-2) *If $deg(u) \geq 3$ and $deg_v(u) \geq 3$, it is zero.*
(4-3) *If $deg(u) \leq 2$, it is $1/2$.*

We say that a vertex v *dominates* vertices adjacent to v if OPT selects v. We also say that v *dominates* v itself. If a vertex u arrives at a vertex v, (v, u) denotes the edge between v and u. Suppose that a vertex u arrives at a vertex v. Also, suppose that u is dominated by a vertex in U and v is dominated by a vertex not in U, in which U is the set of u and all the descendants of u. Then, we say that the edge (v, u) is *free*. We say that a free edge (v, u) is *fixed* if this edge satisfies the following three conditions: (i) $v \neq v_1$, (ii) $deg(u) \geq 3$, and (iii) either $deg(v) = 3$ or $deg(v) = 2$, $deg(v') \geq 3$ and $deg_v(v') \geq 3$, in which $v'(\neq u)$ is the vertex at which v arrives. We say that a vertex triplet (u_1, u_2, u_3) is *good* if the vertices u_1, u_2 and u_3 satisfy the following three conditions: (i) both u_1 and u_3 are adjacent to u_2, (ii) $deg(u_1) = deg(u_2) = deg(u_3) = 3$, and (iii) OPT selects u_1 and u_3.

In the rest of this section, we will show the following lemma.

▶ **Lemma 4.** *There exists an input σ which maximizes $\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)}$ and satisfies the following seven properties.*

(P1) *Any free edge is fixed (Lemmas 7 and 9).*
(P2) *The degree of any vertex is at most three (Lemma 8).*
(P3) *The degree of any vertex selected by OPT is three (Lemma 10).*
(P4) *For any free edge (v, u), OPT does not select v (Lemma 11).*
(P5) *Good vertex triplets are not contained (Lemma 12).*
(P6) *For any free edge (v, u), the degree of v is not two (Lemma 14).*
(P7) *The degree of any vertex is either one or three (Lemma 13).*

This lemma shows that we only have to consider an input satisfying the properties from (P1) to (P7) to evaluate the competitive ratio of RA. It is easy to see that if (P7) holds, both (P2) and (P6) clearly hold. However, we must prove some lemmas including ones about the both properties before showing (P7).

To prove the above lemma and the following lemmas, we give a few definitions about transformations of an input. First, we "divide" an input into two inputs. For an input $\sigma = (V, E, S)$ and a vertex $v \in V$, we define the input $f_1(\sigma, v) = (V_1, E_1, S_1)$ such that $V_1 = V \setminus U$, in which U is the set of v and all the descendants of v, $E_1 = \{\{u, u'\} \in E \mid u, u' \in V_1\}$. That is, (V_1, E_1) is the subgraph of (V, E) induced by V_1, and S_1 is the subsequence of S consisting of all the vertices of V_1 . Also, we define the input $f_2(\sigma, v) = (U, E_2, S_2)$ such that $E_2 = \{\{u, u'\} \in E \mid u, u' \in U\}$, that is, (U, E_2) is the subgraph of (V, E) induced by U, and S_2 is the subsequence of S consisting of all the vertices of U.

Moreover, we "connect" two inputs. For an input $\sigma' = (V', E', S')$, a vertex $v' \in V'$ and an input $\sigma'' = (V'', E'', S'')$, we define $f_3(\sigma', v', \sigma'') = (V_3, E_3, S_3)$ such that $V_3 = V' \cup V''$, $E_3 = E' \cup E'' \cup \{\{v', u''_1\}\}$, in which u''_i is the $i(\in [1, n''])$-th vertex in S'' and n'' is the number of vertices in S'', and $S_3 = (u'_1, \dots, u'_{n'}, u''_1, \dots, u''_{n''})$, in which n' is the number of vertices in S' and u'_i is the $i(\in [1, n'])$-th vertex in S'.

▶ **Lemma 5.** *Suppose that the vertex set of an input σ contains two vertices v and u such that the edge (v, u) is free. Then, there exists OPT such that $D_{OPT}(f_1(\sigma, u)) \cup D_{OPT}(f_2(\sigma, u)) = D_{OPT}(\sigma)$.*

▶ **Lemma 6.** *Suppose that the graph in an input σ contains a vertex v and OPT selects v. Then, there exists OPT such that $D_{OPT}(f_3(\sigma, v, \hat{\sigma})) = D_{OPT}(\sigma)$, in which $\hat{\sigma} = (\{u\}, \varnothing, u)$ and u is a vertex not in the vertex set of the graph in σ.*

▶ **Lemma 7.** *Suppose that the graph of an input σ contains at least one free edge which is not fixed. Then, there exits an input σ' such that (a) any free edge in the graph of σ' is fixed, that is (P1) holds, and (b)* $\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{\mathbb{E}[C_{RA}(\sigma')]}{C_{OPT}(\sigma')}$.

▶ **Lemma 8.** *Suppose that an input σ satisfies the following conditions: (i) the graph in σ contains at least one vertex of degree at least four, and (ii) (P1) holds.*

Then, there exists an input σ' such that (a) the degree of any vertex of the graph in σ' is at most three, that is, (P2) holds, and (b) $\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{\mathbb{E}[C_{RA}(\sigma')]}{C_{OPT}(\sigma')}$.

▶ **Lemma 9.** *Suppose that an input σ satisfies the following conditions: (i) the graph in σ contains at least one free edge which is not fixed, and (ii) (P2) holds.*

Then, there exists an input σ' such that (a) (P1) and (P2) hold, and (b) $\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{\mathbb{E}[C_{RA}(\sigma')]}{C_{OPT}(\sigma')}$.

▶ **Lemma 10.** *Suppose that an input σ satisfies the following conditions: (i) OPT selects at least one vertex of degree at most two, and (ii) (P1) and (P2) hold.*

Then, there exists an input σ' such that (a) the degree of any vertex selected by OPT is three, that is, (P3) holds, (b) (P1) and (P2) hold, and (c) $\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{\mathbb{E}[C_{RA}(\sigma')]}{C_{OPT}(\sigma')}$.

▶ **Lemma 11.** *Suppose that an input σ satisfies the following conditions: (i) there exists at least one free edge (v, u) such that OPT selects v, and (ii) (P1), (P2) and (P3) hold.*

Then, there exists an input σ' such that (a) for any free edge (v, u), OPT does not select v, that is, (P4) holds, (b) (P1), (P2) and (P3) hold, and (c) $\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{\mathbb{E}[C_{RA}(\sigma')]}{C_{OPT}(\sigma')}$.

▶ **Lemma 12.** *Suppose that an input σ satisfies the following conditions: (i) the graph in σ contains at least one good vertex triplet, and (ii) the properties from (P1) to (P4) inclusive hold.*

Then, there exists an input σ' such that (a) the graph in σ' contains no good vertex triplets, that is, (P5) holds, (b) the properties from (P1) to (P4) inclusive hold, and (c) $\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{\mathbb{E}[C_{RA}(\sigma')]}{C_{OPT}(\sigma')}$.

▶ **Lemma 13.** *Consider the graph in an input satisfying the properties from (P1) to (P6) inclusive. Then, the degree of any vertex is one or three.*

▶ **Lemma 14.** *Suppose that an input σ satisfies the following conditions: (i) there exists at least one free edge (v, u) such that $deg(v) = 2$, and (ii) the properties from (P1) to (P5) inclusive hold.*

Then, there exists an input σ' such that (a) for any free edge (v, u), $deg(v) = 3$, (b) the properties from (P1) to (P5) inclusive hold, and (c) $\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{\mathbb{E}[C_{RA}(\sigma')]}{C_{OPT}(\sigma')}$.

Now we can show Lemma 4 using Lemmas 13 and 14. In the next section, we analyze only inputs satisfying the properties from (P1) through (P7).

4.3 Analysis of RA

We assign a positive integer to each vertex of a given tree according to the below routine. We call the set of all the vertices with the same assigned value a *block*. All the vertices in a block are on a path of at most length three. We obtain the competitive ratio of RA by evaluating the costs of RA and OPT for each block. For a vertex v, $N(v)$ denotes the set of vertices adjacent to v. That is, $N(v) = \{u \mid \{v, u\} \in E\}$, in which E is the set of all the edges of a given graph.

```
Algorithm 2: BlockRoutine
    Step 1: $\ell := 0$ and $U := \{v_i \mid i = 1, \dots, n\}$, in which $n$ is the number of all the vertices of a given graph.
    Step 2: $\ell := \ell + 1$. If $U = \varnothing$, then finish. Otherwise, $i_1 := \min\{i \mid v_i \in U\}$, $U := U \setminus \{v_{i_1}\}$ and assign $\ell$ to the vertex $v_{i_1}$.
    Step 3: If $U \cap N(v_{i_1}) = \varnothing$, then go to Step 2. Otherwise, $i_2 := \min\{i \mid v_i \in U \cap N(v_{i_1})\}$, $U := U \setminus \{v_{i_2}\}$ and assign $\ell$ to the vertex $v_{i_2}$.
    Step 4: $U' := U \cap \{N(v_{i_1}) \cup N(v_{i_2})\}$. If $U' = \varnothing$, then go to Step 2. Otherwise, $i_3 := \min\{i \mid v_i \in U'\}$, $U := U \setminus \{v_{i_3}\}$, assign $\ell$ to the vertex $v_{i_3}$ and go to Step 2.
```

▶ **Lemma 15.** *The number of all the vertices of a given tree is at least four.*

By the definition of BlockRoutine, this lemma leads to the fact that at least one vertex in a block is adjacent to a vertex in another block. Also, by (P7) in the previous section, the degree of a vertex is one or three, and hence blocks which a given graph can contain are classified into the following four categories: A ***B_1-block*** is a set consisting of one vertex u_1 such that $deg(u_1) = 1$. The following three blocks are sets consisting of three vertices u_1, u_2 and u_3. Suppose that both u_1 and u_3 are adjacent to u_2. A ***B_2-block*** is a set such that $deg(u_1) = 3$, $deg(u_2) = 3$ and $deg(u_3) = 1$, a ***B_3-block*** is a set such that $deg(u_1) = 1$, $deg(u_2) = 3$ and $deg(u_3) = 1$, and a ***B_4-block*** is a set such that $deg(u_1) = 3$, $deg(u_2) = 3$ and $deg(u_3) = 3$.

For each block, we discuss vertices selected by OPT and classify B_1, B_2, B_3 and B_4 into the following eleven categories. Then the next lemma shows that we only have to consider six categories. u_1, u_2 and u_3 to classify B_i are used in the same definitions as those of u_1, u_2 and u_3 to define B_i.

B_1-blocks are classified into two categories: A ***B_1^0-block*** in which OPT does not select any vertex, and a ***B_1^1-block*** in which OPT selects only u_1.

B_2-blocks are classified into two categories: A ***B_2^{010}-block*** in which OPT selects only u_2, and a ***B_2^{110}-block*** in which OPT selects only u_1 and u_2.

B_3-blocks are not classified. OPT selects only u_2 in a B_3-block.

B_4-blocks are classified into six categories: A ***B_4^{000}-block*** in which OPT selects no vertices, a ***B_4^{100}-block*** in which OPT selects only u_1, a ***B_4^{010}-block*** in which OPT selects only u_2, a ***B_4^{110}-block*** in which OPT selects only u_1 and u_2, a ***B_4^{101}-block*** in which OPT selects only u_1 and u_3, and a ***B_4^{111}-block*** in which OPT selects all the vertices.

▶ **Lemma 16.** *An input can contain at most six kinds of blocks: $B_1^0, B_2, B_3, B_4^{000}, B_4^{100}$ and B_4^{010}.*

A B_1-block consists of one vertex v of degree one, and (4) in Lemma 3 shows that the expected cost of RA for v depends on the adjacent vertex u. Then, we classify B_1-blocks into the following two categories in terms of RA: A $B_{1,0}$-block of v such that $deg_v(u) = 3$ and a $B_{1,1}$-block of v such that $deg_v(u) \leq 2$.

▶ **Lemma 17.** *Consider a block without v_1 and then the expected costs of RA are as follows: (i) zero for a $B_{1,0}$-block, (ii) $1/2$ for a $B_{1,1}$-block, (iii) at most $5/2$ for a B_2-block, (iv) at most $3/2$ for a B_3-block and (v) at most three for a B_4-block.*

Next, we evaluate the expected cost of RA for each block with v_1. Since the number of all the vertices of a given graph is at least four, no B_1-block contains v_1 by the definition of BlockRoutine.

▶ **Lemma 18.** *Consider a block with v_1 and then the expected costs of RA are as follows: (i) at most three for a B_2-block, (ii) at most $5/2$ for a B_3-block and (iii) at most three for a B_4-block.*

Let $b_{1,0}, b_{1,1}, b_2, b_3, b_4^{000}, b_4^{100}$ and b_4^{010} denote the numbers of $B_{1,0}$-blocks, $B_{1,1}$-blocks, B_2-blocks, B_3-blocks, B_4^{000}-blocks, B_4^{100}-blocks and B_4^{010}-blocks, respectively. We define $b_4 = b_4^{000} + b_4^{100} + b_4^{010}$.

▶ **Lemma 19.** *If the number of all the vertices of a given graph is at least five,*

$$b_{1,0} \le b_2 + b_4^{100} + b_4^{010} \tag{1}$$

and

$$b_{1,1} \le b_4^{100}. \tag{2}$$

▶ **Lemma 20.** $b_{1,0} + b_{1,1} + b_3 = b_2 + 3b_4 + 2$.

▶ **Theorem 21.** *The competitive ratio of RA is at most $5/2$.*

Proof. First, we consider an input σ of which the number of vertices of a given tree is four. A combination of blocks composing a tree with four vertices consists of one B_1-block C_1 and one B_3-block C_3 by Lemma 16. Since C_1 does not contain v_1 by the definition of BlockRoutine, C_3 contains v_1. Thus, the expected of RA for C_3 is at most $5/2$ by Lemma 18. Let v denote the vertex in C_1, and let u denote the vertex adjacent to v in C_3. $deg_v(u) = 3$ by the definition of B_3-blocks, which means that C_1 is a $B_{1,0}$-block. Thus, the expected cost for C_1 is zero by Lemma 17. By the above argument, $\mathbb{E}[C_{RA}(\sigma)] = 5/2$. On the other hand, OPT clearly selects at least one vertex, that is, $C_{OPT}(\sigma) \ge 1$. Therefore, we have shown the statement of the theorem in the case of a tree with four vertices.

Next, we consider the case in which of a graph with at least five vertices. The expected costs of RA for a B_2-block and a B_3-block with v_1 are greater than those for a B_2-block and a B_3-block without v_1 by $1/2$ and one, respectively, by Lemmas 17 and 18. Also, v_1 does not affect the expected cost for B_4-blocks. Thus, let b_2' and b_3' denote the numbers of B_2-blocks and B_3-blocks with v_1, respectively. By definition, $b_2', b_3' \in \{0, 1\}$ and $b_2' + b_3' \le 1$. Then, using Lemma 17, we have

$$\mathbb{E}[C_{RA}(\sigma)] \le b_{1,1}/2 + 5b_2/2 + 3b_3/2 + 3b_4 + b_2'/2 + b_3' \le b_{1,1}/2 + 5b_2/2 + 3b_3/2 + 3b_4 + 1.$$

By the definitions of blocks, we have

$$C_{OPT}(\sigma) = b_2 + b_3 + b_4^{100} + b_4^{010}.$$

By the inequality and the equality, we have

$$\begin{aligned}
\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} &\le \frac{b_{1,1}/2 + 5b_2/2 + 3b_3/2 + 3b_4 + 1}{b_2 + b_3 + b_4^{100} + b_4^{010}} \\
&= \frac{-3b_{1,0}/2 - b_{1,1} + 4b_2 + 15b_4/2 + 4}{-b_{1,0} - b_{1,1} + 2b_2 + 3b_4 + b_4^{100} + b_4^{010} + 2} \quad \text{(by the substitution for } b_3 \text{ by Lemma 20)} \\
&\le \frac{-3b_{1,0}/2 + 4b_2 + 15b_4/2 - b_4^{100} + 4}{-b_{1,0} + 2b_2 + 3b_4 + b_4^{010} + 2} \quad \text{(by the substitution for } b_{1,1} \text{ by Eq. (2))} \\
&\le \frac{-5b_{1,0}/2 + 5b_2 + 15b_4/2 + b_4^{010} + 4}{-b_{1,0} + 2b_2 + 3b_4 + b_4^{010} + 2} \quad \text{(by the substitution for } b_4^{100} \text{ by Eq. (1))} \\
&= \frac{5}{2} \cdot \frac{-b_{1,0} + 2b_2 + 3b_4 + 2b_4^{010}/5 + 8/5}{-b_{1,0} + 2b_2 + 3b_4 + b_4^{010} + 2} < \frac{5}{2}.
\end{aligned}$$

◀

Our analysis of RA is exact by the following theorem.

▶ **Theorem 22.** *The competitive ratio of RA is at least $5/2$.*

5 Randomized Lower Bound

▶ **Lemma 23.** *Consider a randomized online algorithm RON. Suppose that a vertex v arrives at a vertex u. Let $p_u(p_v)$ denote the probability that $u \in D_{RON}(v)(v \in D_{RON}(v))$. Then, $p_u + p_v \geq 1$.*

Proof. Let p'_u be the probability that $u \in D_{RON}(u)$. Since RON's selection is irrevocable, the probability that $u \in D_{RON}(u)$ and $u \in D_{RON}(v)$ is greater than or equal to p'_u (Fact (a)). Next, we consider the case in which $u \notin D_{RON}(u)$. RON must select u or v to construct a dominating set immediately after v is revealed. Thus, the probability that either $u \in D_{RON}(v)$ or $v \in D_{RON}(v)$ is one. By the definition of p'_u, the probability that $u \notin D_{RON}(u)$ is $1 - p'_u$. Hence, the probability that both $u \notin D_{RON}(u)$ and either $u \in D_{RON}(v)$ or $v \in D_{RON}(v)$ is at least $1 - p'_u$. This probability together with Fact (a) shows that $p_u + p_v \geq p'_u + 1 - p'_u = 1$. ◀

▶ **Theorem 24.** *The competitive ratio of any randomized online algorithm is at least $4/3$.*

Proof. Consider a randomized online algorithm RON for the following input σ. Let m be any positive integer. We sketch an adversary constructing σ. First, the adversary gives a line of $2m$ vertices to ON. Then, for every two consecutive vertices on the line, the adversary determines whether an additional vertex will arrive at one of the two vertices. Specifically, if the probability that RON selects at least one of the two vertices is low, the adversary makes a new vertex arrive at the vertex.

For each $i = 1, 2, \dots, 2m$, the i-th vertex v_i arrives at v_{i-1}. For each $j = 1, 2, \dots, 2m$, let p_j be the probability that $v_j \in D_{RON}(v_{2m})$. Next, for each $\ell = 1, 2, \dots, m$, vertices are revealed after the revelation of v_{2m} in the following two cases.
Case 1 ($\min\{p_{2\ell-1}, p_{2\ell}\} \geq 2/3$): A new vertex does not arrive at either $v_{2\ell-1}$ or $v_{2\ell}$. Since RON's selection is irrevocable, the expected cost of RON for $v_{2\ell-1}$ and $v_{2\ell}$ is at least $2 \cdot 2/3 = 4/3$.
Case 2 ($\min\{p_{2\ell-1}, p_{2\ell}\} < 2/3$): If $p_{2\ell-1} \leq p_{2\ell}$, then define $\ell_1 = 2\ell - 1$ and $\ell_2 = 2\ell$. Otherwise, define $\ell_2 = 2\ell - 1$ and $\ell_1 = 2\ell$. Then, a vertex u_{ℓ_1} arrives at v_{ℓ_1}. By Lemma 23, the probability that $v_{\ell_1} \in D_{RON}(u_{\ell_1})$ or $u_{\ell_1} \in D_{RON}(u_{\ell_1})$ is at least one (Fact (a)). Moreover, $p_{2\ell-1} + p_{2\ell} = p_{\ell_1} + p_{\ell_2} \geq 1$ also holds. Since $p_{\ell_1} < 2/3$ by the condition of Case 2, $p_{\ell_2} \geq 1/3$. Hence, the expected cost for v_{ℓ_1}, v_{ℓ_2} and u_{ℓ_1} is at least $1 + 1/3 = 4/3$.

Let x be the number of such ℓ with applying Case 1. Thus, the number of such ℓ with applying Case 2 is $m - x$. $\mathbb{E}[C_{RON}(\sigma)] \geq 4x/3 + 4(m-x)/3 = 4m/3$ by the above argument.

Next, we consider an offline algorithm OFF to give an upper bound on the cost of OPT. For such ℓ with applying Case 1, OFF selects $v_{2\ell-1}$ and for such ℓ with applying Case 2, selects v_{ℓ_1}. Thus, OFF selects m vertices, and the set of the m selected vertices is clearly a dominating set. By the optimality of OPT, $C_{OPT}(\sigma) \leq C_{OFF}(\sigma) = m$. Therefore, $\mathbb{E}[C_{RON}(\sigma)]/C_{OPT}(\sigma) \geq 4/3$. ◀

6 Conclusions

In this paper, we have conducted research on algorithms for an online variant of the minimum dominating set problem on trees and obtained the following results: First, we have shown that the competitive ratio of any deterministic algorithm is at least 3, which matches the upper bound shown in [4, 2]. Then, we have designed an algorithm whose competitive ratio is exactly 5/2 using randomization. Furthermore, we have shown that the competitive ratio of any randomized algorithm is at least 4/3.

We conclude this paper by providing open questions: (i) Online algorithms for dominating sets on several graph classes have been discussed in [4, 2] and optimal online algorithms have not yet known on some classes. Then, it is interesting to consider online algorithms on other classes in addition to them. (ii) Our algorithm *RA* is the first randomized algorithm for the online dominating set problem on trees and can achieve a competitive ratio smaller than that of any deterministic algorithm. Can we also obtain a better ratio on other classes using randomization? (iii) The gap between the randomized bounds shown in this paper is still large and thus, it is an obvious open problem to close the gap.

References

1. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
2. J. Boyar, S. J. Eidenbenz, L. M. Favrholdt, M. Kotrbčik, and K. S. Larsen. Online dominating set. In *Proc. of the 15th Scandinavian Symposium and Workshops on Algorithm Theory*, pages 21:1–21:15, 2016.
3. D.-Z. Du and P.-J. Wan. *Connected Dominating Set: Theory and Applications*. Springer, 2013.
4. S. J. Eidenbenz. Online dominating set and variations on restricted graph classes. Technical report, Institute of Theoretical Computer Science, ETH Zürich, 2002.
5. F. V. Fomin, D. Kratsch, and G. J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *Proc. of the 30th international conference on Graph-Theoretic Concepts in Computer Science*, pages 245–256, 2004.
6. F.V. Fomin, F. Grandoni, and D. Kratsch. Some new techniques in design and analysis of exact (exponential) algorithms. In *Bulletin of the EATCS*, pages 47–77, 2005.
7. W. Goddard and M. A. Henning. Independent domination in graphs: A survey and recent results. *Discrete Mathematics*, 313(7):839–854, 2013.
8. F. Grandoni. Independent domination in graphs: A survey and recent results. *Journal of Discrete Algorithms*, 4(2):209–214, 2006.
9. M. Henning and A. Yao. *Total Domination in Graphs*. Springer, 2013.
10. Y. Iwata. A faster algorithm for dominating set analyzed by the potential method. In *Proc. of the 6th international conference on Parameterized and Exact Computation*, pages 41–54, 2011.
11. G. H. King and W. G. Tzeng. On-line algorithms for the dominating set problem. *Information Processing Letters*, 61:11–14, 1997.
12. F. Plastria. Static competitive facility location: An overview of optimisation approaches. *European Journal of Operational Research*, 129(3):461–470, 2001.
13. I. Schiermeyer. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 156(17):3291–3297, 2008.
14. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
15. J. M. M. van Rooij and H. L. Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164, 2011.
16. J. Y. Yu and P. H. J. Chong. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, 7(1):32–48, 2005.