Improved Bounds for Online Dominating Sets of Trees

Koji M. Kobayashi

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 $4/3 \approx 1.333$.

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 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, 15].
The cost of an algorithm $ALG$ for an input $\sigma$ is the size of a dominating set constructed by $ALG$ for $\sigma$, 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 $\sigma$, $C_{ON}(\sigma) \leq cC_{OPT}(\sigma)$, in which $OPT$ is an optimal offline algorithm for $\sigma$. 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 $\sqrt{n}$ 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, 9, 14, 13, 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. [3,8] and [7], 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 \( \sigma \) is the number of vertices in \( U \) at the end of \( \sigma \), 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 \( \sigma \), \( C_{ON}(\sigma) \leq c C_{OPT}(\sigma) \). If \( ON \) is a randomized online algorithm, then the expected cost of \( ON \) is used, which is denoted by \( E[C_{ON}(\sigma)] \). If for any input \( \sigma \), \( E[C_{ON}(\sigma)] \leq c C_{OPT}(\sigma) \), 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 \( \sigma \). We will omit \( \sigma \) 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

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 some 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*-set. Otherwise, a *T*-set such that ℓ modulo 3 = i is called a *T*-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*-sets and at least zero *T*-set, (2) a set of one *T*-set and at least zero *T*-set, (3) a set of one *T*-set, at most one *T*-set and at least zero *T*-set, and (4) a set of sufficiently many *T*-sets and at most one *T*-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*-set and then does one *T*-set (Further, *ON* may also construct *T*-sets). In this case, the routine partitions the *T*-set into a vertex *u*, a *T*-set and a *T*-set (Fig. 2). This *T*-set and all the *T*-sets in (5) except for the partitioned *T*-set compose a set of *T*-sets of (1). Then, the routine finishes constructing a subtree from the current base vertex, whose cost ratio is three, and appoints *u* as a new base vertex. One *T*-set, which is constructed from the above partition of the *T*-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 3.1* For any ε > 0, the competitive ratio of any deterministic online algorithm is at least 3 − ε.

3.2 Proof of Theorem 3.1

Consider a deterministic online algorithm *ON* for the input *σ*, which will be defined below. Without loss of generality, we assume that *ON* does not select any vertex when a vertex arrives at a selected vertex. Moreover, we assume that when a vertex arrives at a vertex which is not selected, *ON* selects only one of the two vertices. First we define a routine to construct a subinput. When this routine is executed, it takes a vertex which has already been revealed to *ON* as a parameter. We call such a vertex a *base vertex* (whose formal definition will be given later). Let us outline this routine. The routine constructs a subtree with at most two leaves from a base vertex, which is the root of this subtree. The routine continues increasing inner vertices of the subtree by the time *ON*
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.
selects a newly revealed vertex. When $ON$ selects it, the routine sets it to be a leaf and reveals another leaf. $MaxLength$ is a sufficiently large positive integer such that $MaxLength$ modulo 3 = 2, which is used in the routine.

| Subtree Routine (vertex $v$) |
|-------------------------------|
| Step T1: $u_0 := v$ and $j := 0.$ |
| Step T2: If $j = MaxLength$, then finish. $j := j + 1.$ Step T3: A vertex $u_j$ arrives at a vertex $u_{j-1}$. Execute one of the following two cases. |
| Case T3.1 ($ON$ selects $u_j$): A vertex $u'_{j-1}$ arrives at $u_{j-1}$, and finish. |
| Case T3.2 ($ON$ does not select $u_j$): Go to Step T2. |

Note that $ON$ selects either $u_{j-1}$ or $u'_{j-1}$ in Case T3.1 because any online algorithm must select at least one of two consecutive vertices. Also, note that $ON$ selects $u_{j-1}$ if it is not selected immediately before the execution of Case T3.2 because $D_{ON}(u_j)$ must contain either $u_{j-1}$ or $u_j$. ($u_0$ is already selected at the beginning of the routine, which is shown later.) Suppose that the routine is executed with a vertex $v$ as the parameter and $j = t$ holds when the routine finishes. Also, suppose that the routine gives vertices $u_i (i \in [1, t])$ and $u'_{t-1}$ (if any) such that $u_1$ is the $a$-th vertex arriving at $v$, $u_{i+1}$ arrives at $u_i$, and $u'_{t-1}$ arrives at $u_{t-1}$, during the execution of the routine. Then, let us define the vertex set $g(v, a)$ as follows: (i) If for all $i \in [1, t]$, $u_i$ is not a base vertex, then $g(v, a) = \{u_1, u_2, \ldots, u_t, u'_{t-1}\}$. (ii) Suppose that for any $i \in [1, x-1]$, $u_i$ is not a base vertex but $u_x$ is a base vertex. If $x \leq t - 2$, $g(v, a) = \{u_1, u_2, \ldots, u_{x-1}\}$. Otherwise, that is, if $x \geq t - 1$, $g(v, a) = \{u_1, u_2, \ldots, u_{t-1}, u'_{t-1}\}$. ($u'_{t-1}$ is not a base vertex. Thus, $g(v, a)$ does not contain any base vertex. Immediately after the execution of the routine, (i) always holds. $u_2$ in (i) can be set to be a base vertex later and then (ii) can hold. Namely, both $x = 2$ and $t$ modulo 3 = 2 in (ii). These facts will be shown later.) We call this vertex set $g(v, a)$ the $a$-th $T$-set of $v$, and say that $v$ has the $T$-set. Also, we define $\ell(v, a) = \max\{i \mid u_i \in g(v, a)\}$, which is called the length of $g(v, a)$. Let us classify $g(v, a)$ into the following four categories: (i) $g(v, a)$ such that $\ell(v, a) = MaxLength$ is called a $T_3$-set, (ii) $g(v, a)$ such that $\ell(v, a) < MaxLength$ and $\ell(v, a)$ modulo 3 = 0 is called a $T_0$-set, (iii) $g(v, a)$ such that $\ell(v, a) < MaxLength$ and $\ell(v, a)$ modulo 3 = 1 is called a $T_1$-set, and (iv) $g(v, a)$ such that $\ell(v, a) < MaxLength$ and $\ell(v, a)$ modulo 3 = 2 is called a $T_2$-set. Next, we evaluate the cost of $ON$ for $g(v, a)$, that is, the number of vertices in $D_{ON} \cap g(v, a)$.  

**Cost for a $T_3$-set**: $\ell(v, a) = MaxLength$ and $D_{ON}(u_t)$ (namely, $D_{ON}$) contains $u_t$ ($i = 1, 2, \ldots, (\ell(v, a) - 2)$ and either $u_{\ell(v, a)-1}$ or $u'_{\ell(v, a)-1}$. Thus, the number of vertices selected by $ON$ in $g(v, a)$ is $MaxLength - 1$.

**Costs for a $T_0$-set, a $T_1$-set, and a $T_2$-set**: $D_{ON}$ contains $u_t$ ($i = 1, 2, \ldots, (\ell(v, a) - 2, \ell(v, a)$). $D_{ON}$ also contains either $u_{\ell(v, a)-1}$ or $u'_{\ell(v, a)-1}$. Thus, the number of vertices selected by $ON$ in $g(v, a)$ is $\ell(v, a)$.

We define the following offline algorithm OFF to give an upper bound on the cost of OPT. Roughly speaking, OFF selects one vertex every consecutive three vertices starting with $u'_{\ell(v, a)-1}$ to deal with $u'_{\ell(v, a)-1}$. Moreover, a $T$-set does not contain a base vertex $v_1 (= u_0)$, but OFF selects $v$ to cover $u_1$ for a $T_1$-set. OFF’s cost for a $T_1$-set includes the cost for $v$.

**Cost for a $T_0$-set**: OFF selects $\ell(v, a)/3$ vertices of $u_2, u_5, u_8, \ldots, u_{\ell(v, a)-1}$.

**Cost for a $T_1$-set**: OFF selects $(\ell(v, a) + 2)/3$ vertices of $u_0, u_3, u_6, \ldots, u_{\ell(v, a)-1}$.

**Cost for a $T_2$-set**: OFF selects $(\ell(v, a) + 1)/3$ vertices of $u_1, u_4, \ldots, u_{\ell(v, a)-1}$.

**Cost for a $T_3$-set**: OFF selects $(\ell(v, a) + 1)/3$ vertices of $u_1, u_4, \ldots, u_{\ell(v, a)-1}$ (OFF’s selection
begins with \( u_1 \) for \( u_0 \), and \( \ell(v, a) = \text{MaxLength} \) such that \( \text{MaxLength} \) modulo 3 = 2 by definition).

\( \text{ON} \) selects all base vertices, which is shown later, while \( \text{OFF} \) selects a base vertex if its \( T \)-sets contain a \( T_1 \)-set or are all \( T_0 \)-sets. Then, we introduce another routine \text{TreeRoutine} to construct the input \( \sigma \). This routine uses \text{SubtreeRoutine} as a subroutine. Now we formally define a base vertex. A base vertex denotes a vertex set to be the value \( \text{BaseVtx} \) in the routine. (This is because this definition also applies to a vertex \( u_2 \) when the routine finishes in Case 3.3.2.) At the beginning, the routine sets \( v_1 \) to be the first base vertex and calls \text{SubtreeRoutine}. Obtaining a \( T \)-set by the call, the routine sets a vertex in the \( T \)-set to be \( \text{BaseVtx} \) if it executes Case 3.3.2. Thus, more than one vertices can be base vertices. After the definition of the routine, we will prove that all the base vertices except at most one are classified into the following four categories: (1) a base vertex with two \( T_1 \)-sets and at least two \( T_0 \)-sets, (2) a base vertex with one \( T_2 \)-set and at least zero \( T_0 \)-sets, (3) a base vertex with one \( T_3 \)-set, at most one \( T_1 \)-set and at most zero \( T_0 \)-sets, and (4) a base vertex with sufficiently many \( T_0 \)-sets and at most one \( T_1 \)-set. Moreover, we will evaluate the cost for each base vertex with its \( T \)-sets and prove that each cost ratio of \( \text{ON} \) to \( \text{OFF} \) is at least three.

Now we define the routine, which uses \( \text{MaxT}_0 \) and \( \text{MaxT}_1 \) as sufficiently large positive integers.

\textbf{TreeRoutine}

\textbf{Initialize:} \( \text{Count} := 1 \). \( v_1 \) is revealed and \( \text{BaseVtx} := v_1 \).

\textbf{Step 1:} \( \text{CntT}_0 := 0 \) and \( \text{CntT}_1 := 0 \).

\textbf{Step 2:}

Call \( \text{SubtreeRoutine}(\text{BaseVtx}) \). Suppose that it gives a \( T \)-set \( g(v, a) \), in which \( v = \text{BaseVtx} \).

\textbf{Step 3:} Execute one of the following four cases.

\textbf{Case 3.1} \( (g(v, a) \text{ is a } T_0 \text{-set}) \) \( \text{CntT}_0 := \text{CntT}_0 + 1 \).

\textbf{Case 3.1.1} \( (\text{CntT}_0 < \text{MaxT}_0) \) : Go to Step 2.

\textbf{Case 3.1.2} \( (\text{CntT}_0 = \text{MaxT}_0) \) : Finish.

\textbf{Case 3.2} \( (g(v, a) \text{ is a } T_1 \text{-set}) \) \( \text{CntT}_0 := \text{CntT}_0 + 1 \).

\textbf{Case 3.2.1} \( (\text{CntT}_1 = 0) \) \( \text{CntT}_1 := 1 \) and go to Step 2.

\textbf{Case 3.2.2} \( (\text{CntT}_1 = 1) \) : Finish.

\textbf{Case 3.3} \( (g(v, a) \text{ is a } T_2 \text{-set}) \):

\textbf{Case 3.3.1} \( (\text{CntT}_1 = 0) \) : Finish.

\textbf{Case 3.3.2} \( (\text{CntT}_1 = 1) \) :

Let \( u_1 \in g(v, a) \) be the vertex arriving at \( \text{BaseVtx} \) and let \( u_2 \in g(v, a) \) be the vertex which first arrived at \( u_1 \). \( g(v, a) \) is a \( T_2 \)-set and hence \( u_2 \) exists (Fig. 2). \( \text{BaseVtx} := u_2 \). If \( \text{Count} = \text{MaxT}_1 \), then finish. Otherwise, \( \text{Count} := \text{Count} + 1 \) and go to Step 1.

\textbf{Case 3.4} \( (g(v, a) \text{ is a } T_3 \text{-set}) \) : Finish.

\( \text{ON} \) must select \( v_1 \). Thus, when \text{SubtreeRoutine} is called with \( \text{BaseVtx} = v_1 \) in Step 2, this subroutine executes Case 3.2 for the first execution of Step T3. Now we analyze the routine. In Case 3.3.2, a new base vertex is set and the value of \( \text{Count} \) increases. Let \( y \) be the value of \( \text{Count} \) when the routine finishes. Then, if the routine finishes in Case 3.1.2, 3.2.2, 3.3.1, or 3.4, then \( y < \text{MaxT}_1 \) and there exist \( y \) base vertices. Otherwise, that is, if it finishes in Case 3.3.2, then \( y = \text{MaxT}_1 \) and there exist \( y + 1 \) base vertices. Let us evaluate the costs for the \( z \)-th base vertex \( v \) and \( T \)-sets of \( v \).

\( (1) \) : First, suppose that \( z \in [1, y - 1] \). Vertices composing \( T \)-sets of \( v \) are as follows: at most two vertices in a \( T \)-set \( g(v, a) \), which is a \( T_2 \)-set, constructed immediately before the execution of
Figure 2: Execution example of Case 3.2.2. \( v \) is a base vertex when Case 3.2.2 is executed. \( \text{SUBTREEROUTINE} \) is executed with \( v \) and constructs a \( T_2 \)-set \( g(v, a) \). Since \( u_2 \) is set to be a new base vertex in Case 3.3.2, the original \( T_2 \)-set is divided into one \( T_1 \)-set \( g(v, a) \) whose length is one and one \( T_0 \)-set \( g(u_2, 1) \). Thus, if \( v \) is the \( \text{Count} \)-th base vertex, then \( u_2 \) is the \( \text{Count+1} \)-st base vertex. Also, \( u_1 \) becomes a \( T \)-set of \( v \) and all the descendants of \( u_2 \) become a \( T_0 \)-set of \( u_2 \).

Case 3.3.2, and all the vertices in \( T \)-sets constructed at the executions of Cases 3.1.1 and 3.2.1 before the execution of Case 3.3.2. Cases 3.1.1 and 3.2.1 are executed at least zero times and once, respectively. In Case 3.3.2, the routine sets a vertex \( u_2 \) in \( g(v, a) \) to be a new base vertex. The original \( T \)-set \( g(v, a) \) is divided into two \( T \)-sets: a \( T_1 \)-set \( g(v, a) \) of \( u_1 \) (if any) and a \( T_0 \)-set \( g(u_2, 1) \) of the rest of the vertices in the original \( g(v, a) \) (see Fig. 2). \( \text{ON} \) must select the first revealed vertex \( v_1 \) and also selects \( u_2 \) by the definition of \( T_2 \)-sets. Hence, \( \text{ON} \) selects all base vertices.

By organizing the above argument, \( T \)-sets of \( v \) are as follows: one \( T_0 \)-set is constructed every time Case 3.1.1 is executed, one \( T_1 \)-set is constructed when Case 3.2.1 is executed, and one \( T_1 \)-set is constructed when Case 3.3.2 is executed.

That is, \( v \) has two \( T_1 \)-sets and \( x - 2 (\geq 0) \) \( T_0 \)-sets. Thus, by the argument about the costs of \( T \)-sets, which were discussed before the routine, the total cost for these \( T \)-sets and \( v \) is as follows: Suppose that the lengths of the two \( T_1 \)-sets are \( t_1 \) and \( t_2 \), respectively and those of the \( x - 2 \) \( T_0 \)-sets are \( t_3, t_4, \ldots, t_x \), respectively. Then, the cost of \( \text{ON} \) is \( \sum_{i=1}^{x} t_i + 1 \), in which the second term follows from the base vertex \( v \). On the other hand, the total cost of \( \text{OFF} \) for the two \( T_1 \)-sets and \( v \) is \((t_1 + 2)/3 + (t_2 - 1)/3\) (\( v \) is selected only once and the former term is greater than the latter by one). The cost for \( x \) \( T_0 \)-sets is \( \sum_{i=3}^{x} t_i/3 \). Hence, the total cost of \( \text{OFF} \) for these \( T \)-sets is \((\sum_{i=1}^{x} t_i + 1)/3\).

(2): Next, suppose that \( z = y \). We consider the cases finishing the routine: Cases 3.1.2, 3.2.2, 3.3.1, 3.3.2, and 3.4.

(2-1) Case 3.2.2: \( \text{CntT}_1 \) used in the routine denotes the number of \( T_1 \)-sets of \( v \), and thus \( v \) has two \( T_1 \)-sets when the routine finished in Case 3.2.2. The routine can execute Case 3.1.1 several times before the finish. Moreover, as mentioned in (1), if \( z \geq 2 \), the routine executes Case 3.2.2 when \( z = y - 1 \), and one \( T_0 \)-set of \( v \), which is called \( u_2 \) in Case 3.2.2, is constructed.

By the above argument, \( v \) has two \( T_1 \)-sets and can have several \( T_0 \)-sets. Therefore, the cost for these \( T \)-sets and \( v \) is obtained in the same way as that in (1).

(2-2) Case 3.3.1: \( v \) has one \( T_2 \)-set and several \( T_0 \)-sets similarly to (2-1). Suppose that the length of the \( T_2 \)-set is \( t_1 \) and those of the \( x - 1 (\geq 0) \) \( T_0 \)-sets are \( t_2, \ldots, t_x \), respectively. Then, the cost of \( \text{ON} \) is \( \sum_{i=1}^{x} t_i + 1 \). On the other hand, the cost of \( \text{OFF} \) for \( v \) and the \( T_2 \)-set is \((t_1 + 1)/3\), and that for the \( x - 1 \) \( T_0 \)-sets is \( \sum_{i=3}^{x} t_i/3 \). Hence, the cost of \( \text{OFF} \) is \((\sum_{i=1}^{x} t_i + 1)/3\).

(2-3) Case 3.4: \( v \) has two \( T_3 \)-sets and \( x - 1 (\geq 0) \) \( T_0 \)-sets. In addition, \( v \) has one \( T_1 \)-set when \( \text{CntT}_1 = 1 \) at the finish of the routine (\( \text{CntT}_1 \) is set to be zero or one). First, we consider the case
in which there does not exist such $T_1$-set, that is, $\text{Cnt}T_1 = 0$ at the finish of the routine. Suppose that the length of the $T_3$-set is $t_1 (= \text{MaxLength})$ and those of the $x - 1 T_0$-sets are $t_2, \ldots, t_x$, respectively. Then, the cost of $\text{ON}$ is $1 + t_1 - 1 + \sum_{i=2}^{x} t_i = \sum_{i=1}^{x} t_i$. The cost of $\text{OFF}$ is at most $(t_1 + 1)/3 + \sum_{i=2}^{x} t_i/3$, in which $+1$ follows from $v$. Next, we consider the case in which there exists one $T_1$-set. Suppose that the length of the $T_1$-set is $t_0$. Then, the costs of $\text{ON}$ and $\text{OFF}$ are $\sum_{i=0}^{x} t_i$ and at most $(t_1 + 1)/3 + \sum_{i=2}^{x} t_i/3 + (t_0 - 1)/3$, respectively.

(2-4) Case 3.1.2: $v$ has $\text{Max}T_0$ $T_0$-sets. Also, $v$ has one $T_1$-set if $\text{Cnt}T_1 = 1$ at the finish of the routine. We consider the case in which $\text{Cnt}T_1 = 0$ at the finish. Suppose that $x = \text{Max}T_0$ and the lengths of the $x$ $T_0$-sets are $t_1, \ldots, t_x$, respectively. Then, the costs of $\text{ON}$ and $\text{OFF}$ are $\sum_{i=1}^{x} t_i + 1$ and at most $\sum_{i=1}^{x} t_i/3 + 1$, respectively. We next consider the case in which $\text{Cnt}T_1 = 1$ at the finish. Suppose that the length of the $T_1$-set is $t_0$. Then, the costs of $\text{ON}$ and $\text{OFF}$ are $\sum_{i=0}^{x} t_i + 1$ and at most $\sum_{i=1}^{x} t_i/3 + 1 + (t_0 - 1)/3$, respectively.

(2-5) Case 3.3.2: As mentioned in (1), $v$ has two $T_1$-sets and can have several $T_0$-sets. The costs of $\text{ON}$ and $\text{OFF}$ for this case are obtained in the same way as those in (1) and (2-1). Moreover, we must consider the $z + 1 = y + 1$-st base vertex $u_2$ and its $T$-sets. $u_2$ has one $T_0$-set. Then, the costs of $\text{ON}$ and $\text{OFF}$ are obtained in the same way as those in (2-4). However, $u_2$ has just one $T_0$-set. The cost ratio of $\text{ON}$ to $\text{OFF}$ for $u_2$ and its $T$-sets can be quite smaller than three. However, note that there exist $\text{Max}T_1$ base vertices with two $T_1$-sets, and $\text{Max}T_1$ is a sufficiently large integer.

Now, we are ready to evaluate the competitive ratio of $\text{ON}$. As mentioned above, the routine executes Case 3.3.2 $y - 1$ times and finishes in Case 3.1.2, 3.2.2, 3.3.1, 3.3.2 or 3.4. If it finishes in Case 3.1.2, 3.2.2, 3.3.1 or 3.4, then there exist $y$ base vertices. Otherwise, that is, if it finished in Case 3.3.2, then there exist $y + 1$ base vertices. By the above argument, we obtain the following results.

(2-1) Case 3.2.2: $C_{\text{ON}}(\sigma)/C_{\text{OFF}}(\sigma) = 3$.

(2-2) Case 3.3.1: $C_{\text{ON}}(\sigma)/C_{\text{OFF}}(\sigma) = 3$.

(2-3) Case 3.4: $C_{\text{ON}}(\sigma)/C_{\text{OFF}}(\sigma) = 3 - \epsilon$, in which $\epsilon > 0$ approaches zero as the value $\text{MaxLength}$ in the subroutine becomes sufficiently large.

(2-4) Case 3.1.2: $C_{\text{ON}}(\sigma)/C_{\text{OFF}}(\sigma) = 3 - \epsilon'$, in which $\epsilon' > 0$ approaches zero as the value $\text{Max}T_0$ in the routine becomes sufficiently large.

(2-5) Case 3.3.2: $C_{\text{ON}}(\sigma)/C_{\text{OFF}}(\sigma) = 3 - \epsilon''$, in which $\epsilon'' > 0$ approaches zero as the value $\text{Max}T_1$ in the routine becomes sufficiently large.

We have shown that the statement of the theorem is true.

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.

9
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 ($p(v_i) \mod 2 = 0$): Select $u$.
  - Case 2.2.2 ($p(v_i) \mod 2 = 1$): 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). Thus, the set of vertices selected by $B$ is also a dominating set at any time.

Algorithm B

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 ($p(v_i) \mod 2 = 0$): Select $u$.
  - Case 2.2.2 ($p(v_i) \mod 2 = 1$): Select $v_i$.

4.2 Basic Properties of $RA$

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

Lemma 4.1 The following properties hold for a vertex $v$:

1. If $v = v_1$, $v \in D_A$ and $v \in D_B$.
2. If $\deg(v) \geq 3$, $v \in D_A$ and $v \in D_B$.
3. Suppose that $\deg(v) = 2$.
   - (3-e) If $p(v) \mod 2 = 0$, $v \notin D_A$ and $v \in D_B$.
   - (3-o) If $p(v) \mod 2 = 1$, $v \in D_A$ 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 $p(v) \mod 2 = 0$, $v \notin D_A$ and $v \in D_B$.
   - (4-1-o) If $p(v) \mod 2 = 1$, $v \in D_A$ 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 $p(v) \mod 2 = 0$, $v \notin D_A$ and $v \in D_B$.
   - (4-3-o) If $p(v) \mod 2 = 1$, $v \in D_A$ and $v \notin D_B$.

Proof. To show each property in the statement of this lemma, we prove the following properties
for each $j \leq i$ by induction on the number of revealed vertices. Note that if we have proven them, then each property in the statement is satisfied when $i$ is the number of the vertices of the tree in an input, that is, $v_i$ is the final revealed vertex.

1. If $j = 1$, $v_1 \in D_A(v_1)$ and $v_1 \in D_B(v_1)$.
   Suppose that $j \geq 2$.
2. If $\deg_{v_1}(v_j) \geq 3$, $v_j \in D_A(v_1)$ and $v_j \in D_B(v_1)$.
3. Suppose that $\deg_{v_1}(v_j) = 2$.
   (3-e) If $p(v_j)$ modulo 2 = 0, $v_j \notin D_A(v_1)$ and $v_j \notin D_B(v_1)$.
4. Suppose that $\deg_{v_1}(v_j) = 1$. Let $v_j$ be the vertex adjacent to $v_1$.
   (4-1) Suppose that $\deg_{v_1}(\hat{u}) \geq 3$ and $\deg_{v_1}(\hat{u}) \leq 2$.
   (4-1-e) If $p(v_j)$ modulo 2 = 0, $v_j \notin D_A(v_1)$ and $v_j \notin D_B(v_1)$.
   (4-1-o) If $p(v_j)$ modulo 2 = 1, $v_j \in D_A(v_1)$ and $v_j \notin D_B(v_1)$.
5. Suppose that $\deg_{v_1}(v_j) = 2$. Then, we in what follows show that $v_j$ satisfies (4-1) when $v_j$ is revealed. By induction hypothesis. Analogously to the case in which $v_j$ is revealed, only the degree of an adjacent vertex $u$ to $v_j$ changes. Hence, all the revealed vertices except for $u$ and vertices adjacent to $u$ also satisfy the above properties when $i = m$ by the induction hypothesis. Then, we in what follows show that $u$ and a vertex adjacent to $u$ including $v_m$ satisfy the properties. Note that the degree of $u$ may affect an adjacent vertex $u''(\neq v_m)$ at only the case (4) of the above properties, that is, the case in which $u'' \neq v_1$ and the degree of $u''$ is one. Hence, $u''$ also satisfies the above properties when $i = m$ in the other cases by the induction hypothesis. First, we consider the case in which $u \neq v_1$. Note that $\deg_{v_m}(u) \geq 2$.
6. Suppose that $\deg_{v_m}(u) \geq 4$. When $v_m$ is revealed, both $A$ and $B$ execute Case 2.1 and select $u$. That is, $u \in D_A(v_m)$ and $u \in D_B(v_m)$. Thus, $u$ satisfies the property (2) because $\deg_{v_m}(u) \geq 3$. On the other hand, $A$ and $B$ do nothing for $v_m$, and thus $v_m \notin D_A(v_m)$ and $v_m \notin D_B(v_m)$. Then, $\deg_{v_m}(v_m) = 1$ and $v_m$ satisfies (4-2). Since the degree of $u''$ is one as mentioned above, $u''$ satisfies one of the properties (4-1) and (4-2) before $v_m$ is revealed by the induction hypothesis. Analogously to $v_m$, $A$ and $B$ do nothing $u''$, and $v_m$ keeps (4-1) or (4-2).
7. Suppose that $\deg_{v_m}(u) = 3$. In this case, $A$ and $B$ execute Case 2.1, and $u \in D_A(v_m)$ and $u \in D_B(v_m)$. Similarly to the case in which $\deg_{v_m}(u) \geq 4$, $u$ and $v_m$ satisfy (2) and (4-2), respectively. Let us consider $u''$ next. Since the degree of $u''$ is one and $u''$ is not $v_1$, $\deg_{v_m}(u) \leq 2$.
   Suppose that $p(v_m)$ modulo 2 = 0. $u'' \notin D_A(v_{m-1})$ and $u'' \in D_B(v_{m-1})$ by (4-3-e) in the induction hypothesis. When $v_m$ is revealed, $A$ and $B$ do nothing for $u''$. Thus, $u'' \notin D_A(v_m)$ and $u'' \in D_B(v_m)$, which means that $u''$ satisfies (4-1-e) since $\deg_{v_m}(u) = 3$ and $\deg_{v_m}(u'') = 1$. If $p(v_m)$ modulo 2 = 1, we can show that $u''$ satisfies (4-1-o) similarly.
8. Suppose that $\deg_{v_m}(u) = 2$. Thus, $\deg_{v_{m-1}}(u) = 1$. We discuss $u''$ in advance. $u''$ is revealed before $u$ by the condition of this case because $u$ is not $v_1$. However, the degree of $u''$ is one, which contradicts that $u''$ is not $v_1$. Hence, there does not exist $u''$ in this case. Let $u''(\neq v_m)$ be a vertex adjacent to $u$. Suppose that $\deg_{v_m}(u') \leq 2$ and $p(v_m)$ modulo 2 = 0. $A$ and $B$ execute Case 2.2.1 by definition when $v_m$ is revealed. Hence, $A$ and $B$ select $u$ and $v_m$, respectively, which indicates
that \(v_m \notin D_A(v_m), v_m \in D_B(v_m)\) and \(u \in D_A(v_m)\). Also, \(p(u) \text{ modulo } 2 = 1\) because \(p(v_m) \text{ modulo } 2 = 0\). Hence, \(u \notin D_B(v_m)\) by (4-3-o) in the induction hypothesis. Thus, \(u \notin D_B(v_m)\). Then, \(v_m\) satisfies (4-3-e) since \(\deg_{v_m}(v_m) = 1\). In addition, \(u\) satisfies (3-o) since \(\deg_{v_m}(u) = 2\). The case in which \(\deg_{v_m}(u') \leq 2\) and \(p(v_m)\) modulo 2 = 1 can be shown similarly. Roughly speaking, \(A\) and \(B\) execute Case 2.2.2 in this case, and \(v_m\) and \(u\) satisfy (4-3-o) and (3-e), respectively.

Suppose that \(\deg_{v_m}(u) = 2, \deg_{v_m}(u') \geq 3\) and \(\deg_u(u') \geq 3\). Also, suppose that \(p(v_m)\) modulo 2 = 0. Both \(A\) and \(B\) execute Case 2.2.1 and \(A\) and \(B\) select \(u\) and \(v_m\), respectively, when \(v_m\) is revealed. Thus, \(v_m \notin D_A(v_m), v_m \in D_B(v_m)\) and \(u \in D_A(v_m)\). Since \(\deg_u(u') \geq 3\), \(u \notin D_B(v_m-1)\) by (4-2) in the induction hypothesis, which directly implies that \(u \notin D_B(v_m)\). Hence, \(v_m\) satisfies (4-3-e) because \(\deg_{v_m}(v_m) = 1\). \(u\) satisfies (3-o) because \(\deg_{v_m}(u) = 2\). We can prove the case in which \(p(v_m)\) modulo 2 = 1, similarly, and hence we omit it.

Suppose that \(\deg_{v_m}(u) = 2, \deg_{v_m}(u') \geq 3\) and \(\deg_u(u') \leq 2\). Then, \(A\) and \(B\) execute Case 2.2.1. Moreover, \(u \notin D_B(v_m-1)\) by (4-1-o) in the induction hypothesis if \(p(v_m)\) modulo 2 = 0. We omit the proof of this case because the situation in this case is quite the same as that in the case in which \(\deg_u(u') \geq 3\) and \(p(v_m)\) modulo 2 = 0. In addition, the case in which \(p(v_m)\) modulo 2 = 1 can be proven in the same way as the case in which \(\deg_u(u') \geq 3\) and \(p(v_m)\) modulo 2 = 1.

Suppose that \(u = v_1\). \(u \in D_A(v_m-1)\) and \(u \in D_B(v_m-1)\) by (1) in the induction hypothesis. Thus, \(u \in D_A(v_m)\) and \(u \in D_B(v_m)\), which means that \(u\) satisfies (1). We omit the proof with respect to \(v_m\) because it can be shown in the same way as the case of \(u \neq v_m\). Finally, let us discuss \(u''\). If \(\deg_{v_m}(u) \neq 2\), we can show the proof in the same way as that in the case in which \(u \neq v_m\). Then, we consider the case in which \(\deg_{v_m}(u) = 2\). That is, \(\deg_{v_m-1}(u) = 1\). Since the degree of \(u''\) is one, both \(u'' \in D_A(v_m-1)\) and \(u'' \notin D_B(v_m-1)\) by (4-3-o) in the induction hypothesis. Also, \(A\) and \(B\) execute Case 2.2.2 when \(v_m\) is revealed, and they do not select \(u''\). Thus, \(u'' \in D_A(v_m)\) and \(u'' \notin D_B(v_m)\). Therefore, \(u''\) satisfies (4-3-o) immediately after the revelation of \(v_m\).

**Lemma 4.2** The expected cost of RA for \(v\) is as follows:

1. If \(v = v_1\), it is one.
2. Suppose that \(v \neq v_1\).
3. If \(\deg(v) = 2\), it is \(1/2\).
4. Suppose that \(\deg(v) = 1\) and \(v\) is adjacent to a vertex \(u\).
   1. If \(\deg(u) \geq 3\) and \(\deg_u(u) \leq 2\), it is \(1/2\).
   2. If \(\deg(u) \geq 3\) and \(\deg_u(u) \geq 3\), it is zero.
5. If \(\deg(v) \leq 2\), it is \(1/2\).

**Proof.** RA chooses \(A\) (\(B\)) with the probability \(1/2\) at the beginning. Thus, the expected cost of RA for a vertex \(v\) is \(1/2\) if either \(v \notin D_A\) and \(v \in D_B\) or \(v \in D_A\) and \(v \notin D_B\). It is easy to see that the expected cost for \(v\) is one if both \(v \in D_A\) and \(v \in D_B\). Also, the expected cost for \(v\) is zero if both \(v \notin D_A\) and \(v \notin D_B\). The above argument with each property in Lemma 4.1 gives each cost in the statement of this lemma.

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 $V_1 = V \setminus \{v\}$, 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 $V_1$ (Fig. 3). 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 $V_3 = V' \cup V''$, $E_3 = E' \cup E'' \cup \{(v', v'_i)\}$, in which $v'_i$ is the $i(\in [1, n''])$-th vertex in $\sigma''$, $n''$ is the number of vertices in $S''$, and $S_3 = (u_1', \ldots, u_{n'}, v''_1, \ldots, v''_{n''})$, in which $n'$ is the number of vertices in $S'$ and $v'_i$ is the $i(\in [1, n''])$-th vertex in $\sigma''$.

![Figure 3: “Dividing” and “connecting” inputs. The left figure shows that we construct the two inputs $f_1(\sigma, v)$ and $f_2(\sigma, v)$ in terms of a vertex $v$ from an input $\sigma$. The right figure shows that we obtain the input $f_3(\sigma', v', \sigma'')$ by adding an input $\sigma''$ to an input $\sigma'$ in terms of a vertex $v'$ in $\sigma'$. $u$ was the first revealed vertex $v_1$ in $\sigma''$.](image)

**Lemma 4.3** Suppose that the vertex set of an input $\sigma$ contains two vertices $v$ and $u$ such that the edge $(v, u)$ is free. Then, there exists $OPT$ such that $D_{OPT}(f_1(\sigma, u)) \cup D_{OPT}(f_2(\sigma, u)) = D_{OPT}(\sigma)$.

**Proof.** Suppose that the vertex set of an input $\sigma$ contains two vertices $v$ and $u$ such that the edge $(v, u)$ is free. We consider an offline algorithm $OFF$ for the input $f_1(\sigma, u)$ and input $f_2(\sigma, u)$ which selects the same vertices as those selected by $OPT$ for $\sigma$. That is, $D_{OFF}(f_1(\sigma, u)) \cup D_{OFF}(f_2(\sigma, u)) = D_{OPT}(\sigma)$, and hence,

$$C_{OFF}(f_1(\sigma, u)) + C_{OFF}(f_2(\sigma, u)) = C_{OPT}(\sigma). \quad (1)$$

We prove that $OFF$ is optimal by contradiction. We assume that $OFF$ is not optimal, that is, there exists an offline algorithm $OPT'$ such that

$$C_{OPT'}(f_1(\sigma, u)) + C_{OPT'}(f_2(\sigma, u)) < C_{OFF}(f_1(\sigma, u)) + C_{OFF}(f_2(\sigma, u)). \quad (2)$$

By the definitions of $f_1(\sigma, u)$ and $f_2(\sigma, u)$, $D_{OPT'}(f_1(\sigma, u)) \cup D_{OPT'}(f_2(\sigma, u))$ is a dominating set of the graph of $\sigma$. However, the size of this set is smaller than that of $D_{OPT}(\sigma)$ by Eqs. (1) and (2), which contradicts the optimality of $OPT$. Therefore, $OFF$ is optimal.

**Lemma 4.4** Suppose that the graph in an input $\sigma$ 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 $\hat{\sigma} = (\{u\}, \emptyset, u)$ and $u$ is a vertex not in the vertex set of the graph in $\sigma$. 

13
Proof. We prove this lemma in a similar way to the proof of Lemma 4.3. Suppose that the graph in the vertex set of the graph in an input \( \sigma \) contains a vertex \( v \) and \( \text{OPT} \) selects \( v \). Let \( \hat{\sigma} = (\{u\}, \emptyset, u) \), in which \( u \) is a vertex not in the graph in \( \sigma \). We consider an offline algorithm \( \text{OFF} \) for the input \( f_3(\sigma, v, \hat{\sigma}) \) which selects the same vertices as those selected by \( \text{OPT} \). That is, \( D_{\text{OFF}}(f_3(\sigma, v, \hat{\sigma})) = D_{\text{OPT}}(\sigma) \), and hence,

\[
C_{\text{OFF}}(f_3(\sigma, v, \hat{\sigma})) = C_{\text{OPT}}(\sigma). \tag{3}
\]

We prove that \( \text{OFF} \) is optimal by contradiction. We assume that \( \text{OFF} \) is not optimal, that is, there exists an offline algorithm \( \text{OPT}' \) such that

\[
C_{\text{OPT}}(f_3(\sigma, v, \hat{\sigma})) < C_{\text{OFF}}(f_3(\sigma, v, \hat{\sigma})). \tag{4}
\]

If \( \text{OPT}' \) does not select \( u \) for \( f_3(\sigma, v, \hat{\sigma}) \), \( D_{\text{OPT}'}(f_3(\sigma, v, \hat{\sigma})) \) is a dominating set of the graph in \( \sigma \). The size of this set is smaller than that of \( D_{\text{OPT}}(\sigma) \) by Eqs. (3) and (4), which contradicts the optimality of \( \text{OPT} \).

Next, we consider the case in which \( \text{OPT}' \) selects \( u \). Suppose that \( \text{OPT}' \) selects \( v \) instead of \( u \). Then, the vertex set is also a dominating set of the graph in \( f_3(\sigma, v, \hat{\sigma}) \). Moreover, it is also a dominating set of the graph in \( \sigma \). Hence, the size of this set is smaller than that of \( D_{\text{OPT}}(\sigma) \) by Eqs. (3) and (4). This fact contradicts the optimality of \( \text{OPT} \), and thus \( \text{OFF} \) is optimal. \( \qed \)

Lemma 4.5 Suppose that the graph of an input \( \sigma \) contains at least one free edge which is not fixed. Then, there exists an input \( \sigma' \) such that (a) any free edge in the graph of \( \sigma' \) is fixed, that is \((P1)\) holds, and (b) \( \frac{\mathbb{E}[C_{\text{RA}}(\sigma)]}{C_{\text{OPT}}(\sigma)} \leq \frac{\mathbb{E}[C_{\text{RA}}(\sigma')]}{C_{\text{OPT}}(\sigma')} \).

Proof. Suppose that the graph in an input \( \sigma \) contains at least one free edge which is not fixed. To prove the lemma, we will prove that we can construct an input \( \sigma'' \) satisfying the following two properties from \( \sigma \): (1) the number of free edges which are not fixed in the graph of \( \sigma'' \) is less than that in the graph of \( \sigma \), and (2) \( \frac{\mathbb{E}[C_{\text{RA}}(\sigma)]}{C_{\text{OPT}}(\sigma)} \leq \frac{\mathbb{E}[C_{\text{RA}}(\sigma'')]}{C_{\text{OPT}}(\sigma'')} \). Proving the existence of such \( \sigma'' \) means that we can construct an input satisfying the conditions (a) and (b) in the statement of this lemma recursively.

Let \( (v, u) \) be a free edge which is not fixed of the graph in \( \sigma \). Define \( \sigma_1 = f_1(\sigma, u) \) and \( \sigma_2 = f_2(\sigma, u) \). By Lemma 4.3,

\[
D_{\text{OPT}}(\sigma_1) \cup D_{\text{OPT}}(\sigma_2) = D_{\text{OPT}}(\sigma). \tag{5}
\]

Thus, if an edge of the graph in \( \sigma_1 \) is free (not free), then that in \( \sigma \) is also free (not free). Also, if a free edge of the graph in \( \sigma_1 \) is fixed (not fixed), then that in \( \sigma \) is also fixed (not fixed). This fact is also true with edges in \( \sigma_2 \). We remove the edge \( (v, u) \) when constructing \( \sigma_1 \) and \( \sigma_2 \), and hence the number of free edges which are not fixed in \( \sigma_1 \) (\( \sigma_2 \)) is less than that in \( \sigma \) by at least one, which implies that \( \sigma_1 \) and \( \sigma_2 \) satisfy the property (1) described above. In what follows, we prove the property (2). By Eq. (5), with respect to the costs of \( \text{OPT} \) for \( \sigma_1 \) and \( \sigma_2 \),

\[
C_{\text{OPT}}(\sigma_1) + C_{\text{OPT}}(\sigma_2) = C_{\text{OPT}}(\sigma). \tag{6}
\]

Next, we evaluate the expected cost of \( \text{RA} \) using Lemma 4.2. The expected cost of \( \text{RA} \) for a vertex depends on its degree and the degree of an adjacent vertex, and thus vertices for which the difference between the expected costs in \( \sigma \) and \( \sigma_1(\sigma_2) \) can arise are only \( v, u \) and their adjacent
vertices. First, we discuss the expected cost for a vertex \( v''(\neq u) \) adjacent to \( v \). In \( \sigma \), if \( \text{deg}(v) \geq 3 \), \( \text{deg}(v'') = 1 \), \( \text{deg}_v(v') = 3 \) and \( \text{deg}_u(v) = 2 \), then the expected cost for \( v'' \) in \( \sigma \) is zero. Also, the expected cost for \( v'' \) in \( \sigma_1 \) is 1/2 because \( \text{deg}_u(v) = 2 \). Otherwise, the expected cost for \( v'' \) does not change. Thus, the expected cost for \( v'' \) does not decrease. In a similar way to the expected cost for \( v'' \), that for a vertex adjacent to \( u \) in \( \sigma_2 \) does not decrease.

Next, we evaluate the cost for \( v \). If \( v = v_1 \), the expected cost for \( v \) is one in either \( \sigma \) or \( \sigma_1 \). That is, it does not change. In the rest of the proof, we consider the case in which \( \text{deg}(v) \geq 2 \) in \( \sigma \) holds, and then let \( v'(\neq u) \) be the vertex adjacent to \( v \).

We consider the case in which \( \text{deg}(v) = 2 \) in \( \sigma \), that is, the degree of \( v \) in \( \sigma_1 \) turns one. The expected cost for \( v \) in \( \sigma \) is 1/2. In \( \sigma \), if either \( \text{deg}(v') \leq 2 \) or \( \text{deg}(v') \geq 3 \) and \( \text{deg}_u(v') \leq 2 \), then the expected cost for \( v \) in \( \sigma_1 \) remains 1/2. If \( \text{deg}(v') \geq 3 \) and \( \text{deg}_u(v') \geq 3 \), then the expected cost in \( \sigma_1 \) becomes zero, that is, it decreases by 1/2.

In the case in which \( \text{deg}(v) = 3 \) in \( \sigma \), the expected cost for \( v \) in \( \sigma \) is one. Since \( \text{deg}(v) = 2 \) in \( \sigma_1 \), the expected cost for \( v \) in \( \sigma \) is 1/2 and decreases by 1/2.

In the case in which \( \text{deg}(v) \geq 4 \) in \( \sigma \), the expected cost for \( v \) in \( \sigma \) is one. \( \text{deg}(v) \geq 3 \) in \( \sigma_1 \) as well, and the expected cost for \( v \) remains one.

Next, we evaluate the cost for \( u \). By definition, \( u \) is the first revealed vertex in \( \sigma_2 \), and thus the expected cost for \( u \) in \( \sigma_2 \) is one. We discuss the cost for \( u \) in \( \sigma \). If \( \text{deg}(u) \leq 2 \) in \( \sigma \), then the expected cost is at most 1/2, which means that the expected cost for \( u \) increases by at least 1/2 in \( \sigma_2 \). If \( \text{deg}(u) \geq 3 \) in \( \sigma \), then the expected cost is one, which means that the expected cost for \( u \) does not change.

By the above argument, the case in which
\[
\mathbb{E}[\text{C}_{RA}(\sigma)] > \mathbb{E}[\text{C}_{RA}(\sigma_1)] + \mathbb{E}[\text{C}_{RA}(\sigma_2)]
\]
can hold is only the case in which the expected cost for \( v \) decreases by 1/2 and that for \( u \) does not change, that is, the case in which \((v,u)\) is fixed. Thus, if \((v,u)\) is not fixed,
\[
\mathbb{E}[\text{C}_{RA}(\sigma)] \leq \mathbb{E}[\text{C}_{RA}(\sigma_1)] + \mathbb{E}[\text{C}_{RA}(\sigma_2)].
\]
This inequality together with Eq. (6) shows
\[
\frac{\mathbb{E}[\text{C}_{RA}(\sigma)]}{\text{C}_{OPT}(\sigma)} \leq \frac{\mathbb{E}[\text{C}_{RA}(\sigma_1)] + \mathbb{E}[\text{C}_{RA}(\sigma_2)]}{\text{C}_{OPT}(\sigma_1) + \text{C}_{OPT}(\sigma_2)} \leq \max \left\{ \frac{\mathbb{E}[\text{C}_{RA}(\sigma_1)]}{\text{C}_{OPT}(\sigma_1)} , \frac{\mathbb{E}[\text{C}_{RA}(\sigma_2)]}{\text{C}_{OPT}(\sigma_2)} \right\},
\]
which proves that either \( \sigma_1 \) or \( \sigma_2 \) satisfies the property (2) described above.

\[
\text{Lemma 4.6} \quad \text{Suppose that an input } \sigma \text{ satisfies the following conditions: (i) the graph in } \sigma \text{ contains at least one vertex of degree at least four, and (ii) (P1) holds.}
\]
\[
\text{Then, there exists an input } \sigma' \text{ such that (a) the degree of any vertex of the graph in } \sigma' \text{ is at most three, that is, (P2) holds, and (b) } \frac{\mathbb{E}[\text{C}_{RA}(\sigma)]}{\text{C}_{OPT}(\sigma)} \leq \frac{\mathbb{E}[\text{C}_{RA}(\sigma')]}{\text{C}_{OPT}(\sigma')}.
\]
\[
\text{Proof.} \quad \text{We prove this lemma in a similar way to the proof of Lemma 4.5. Suppose that an input } \sigma \text{ satisfies the conditions (i) and (ii) in the statement. Let } v \text{ and } u \text{ be vertices in } \sigma \text{ such that } u \text{ arrives at } v, \text{deg}(v) \geq 4, \text{deg}_u(v) = \text{deg}(v) \text{ and the degrees of all the descendants of } u \text{ are at most}
\]

15
three. To prove the lemma, we will prove that we can construct an input $\sigma''$ satisfying the following two properties from $\sigma$: (1) all the vertices of the graph $T''$ in $\sigma''$ are in the vertices of the graph $T$ in $\sigma$, the degree of each vertex except for $v$ of $T''$ is equal to that of $T$, the degree of $v$ in $T''$ is less than that of $T$, and (2) $\frac{E[\text{RA}(\sigma)]}{C_{\text{OPT}}(\sigma)} \leq \frac{E[\text{RA}(\sigma'')]}{C_{\text{OPT}}(\sigma'')}$. We can construct an input satisfying the properties (a) and (b) in the statement recursively by proving the existence of $\sigma''$. In what follows, we discuss such input $\sigma''$ for each of the following three cases: (Case 1) $\text{OPT}$ selects $u$ and only $u$ dominates $v$, (Case 2) $\text{OPT}$ selects $v$ and only $v$ dominates $u$, and (Case 3) the other cases.

**Case 1:** $\text{OPT}$ does not select $v$ by the condition of this case. Since $\deg(v) \geq 4$, there exists another vertex $u'(\neq u)$ arriving at $v$. $u'$ does not dominate $v$ by the condition of this case, and $\text{OPT}$ selects a descendant of $u'$ to dominate $u'$. Thus, the edge $(v, u')$ is free. $\deg(v) \geq 4$ by the definition of $v$, and hence $(v, u')$ is not fixed, which contradicts that $\sigma$ satisfies the property (P1) by the condition (ii) in the statement of the lemma. Therefore, this case does not happen.

**Case 2:** Let $U$ be the set of vertices consisting of $u$ and all the descendants of $u$, and let $U'$ be the set of vertices of all the descendants of $v$ except for $U$. We consider the following two cases: (Case 2-1) the degree of any vertex selected by $\text{OPT}$ in $U'$ is three, and (Case 2-2) the other case, that is, there exists at least one vertex selected by $\text{OPT}$ in $U'$ whose degree is at most two. Define $\sigma_1 = f_1(\sigma, u)$ and $\sigma_2 = f_2(\sigma, u)$.

**Case 2-1:** There exist vertices $v', u' \in U'$ by the conditions of Cases 2 and 2-1 such that $\deg(v') = 3$, $u'$ arrives at $v'$, $\deg_{u'}(v') = 3$, $\deg(u') = 1$ and $\text{OPT}$ selects $u'$. Here, we construct the input $\sigma'_1$ by removing $u'$ from $\sigma_1$, and obtain the input $\sigma_3$ by adding $\sigma_2$ to $\sigma_1$ in terms of $v'$. Formally, define $\sigma'_1 = f_1(\sigma, u')$, $\sigma_2 = f_2(\sigma, u)$, and $\sigma_3 = f_3(\sigma'_1, v', \sigma_2)$ (see Fig. 4). Then, $\sigma_3$ satisfies the above property (1).

To show that the property (2) is satisfied, we first evaluate the cost of $\text{OPT}$ for $\sigma_3$. Let $\text{OFF}$ be an offline algorithm for $\sigma_3$ which selects the same vertices as those selected by $\text{OPT}$ for $\sigma$, that is, $D_{\text{OFF}}(\sigma_3) = D_{\text{OPT}}(\sigma)$. Then, $D_{\text{OPT}}(\sigma)$ is a dominating set of the graph in $\sigma_3$ by the definition of $\sigma_3$. Thus,

$$C_{\text{OFF}}(\sigma_3) = C_{\text{OPT}}(\sigma).$$

(7)

We next consider the expected cost of $\text{RA}$ using Lemma 4.2. The degree of each vertex except for $v$ in $\sigma_3$ is equal to that in $\sigma$. The above input transformations do not affect the expected costs for vertices except for $v, v'$ and their adjacent vertices. The expected cost for $v$ does not change because $\deg(v) \geq 3$ before and after the transformations. Also, $\deg_u(v) = \deg(v) \geq 3$ in $\sigma$ and $\deg_{u'}(v') = \deg(v') = 3$ in $\sigma_3$. Thus, the expected cost for $u$ does not change, either. The expected cost for $u'$ is zero because $\deg_{u'}(v') = 3$ in $\sigma$. Moreover, the transformations do not affect the expected costs of all the vertices adjacent to either $v$, $v'$, or $v''$ except for $u$ and $u'$. Hence,

$$\mathbb{E}[C_{\text{RA}}(\sigma_3)] = \mathbb{E}[C_{\text{RA}}(\sigma)].$$

By this equality and Eq. (7),

$$\frac{\mathbb{E}[C_{\text{RA}}(\sigma)]}{C_{\text{OPT}}(\sigma)} = \frac{\mathbb{E}[C_{\text{RA}}(\sigma_3)]}{C_{\text{OFF}}(\sigma_3)} \leq \frac{\mathbb{E}[C_{\text{RA}}(\sigma_3)]}{C_{\text{OPT}}(\sigma_3)},$$

which implies that $\sigma_3$ satisfies the property (2).

**Case 2-2:** Let $v'' \in U'$ be a vertex selected by $\text{OPT}$ of degree at most two. Then, define $\sigma_3' = f_3(\sigma_1, v'', \sigma_2)$, which satisfies the property (1).

Let us consider the cost of $\text{OPT}$ for $\sigma_3'$. In the same way as Case 2-1, let $\text{OFF}$ be an offline algorithm which selects all the vertices in $D_{\text{OPT}}(\sigma)$. $D_{\text{OPT}}(\sigma)$ is a dominating set of the graph in
σ₃ by the definition of σ′₃. Thus,

$$C_{OFF}(σ′₃) = C_{OPT}(σ).$$  \hspace{1cm} (8)

We consider the expected cost of RA. The degrees of all the vertices except for v and v″ do not change. The expected costs for all the vertices except for v″ do not change by the same argument in Case 2-1. Since the degree of v″ is at least two, the expected cost for v″ does not decrease. Hence,

$$E[C_{RA}(σ′₃)] ≥ E[C_{RA}(σ₃)].$$

By this inequality and Eq. (8),

$$E[C_{RA}(σ₃)] ≤ E[C_{RA}(σ′₃)] ≤ E[C_{RA}(σ′₃)].$$

Thus, σ′₃ satisfies the property (2).

(Case 3): (v, u) is free because v (u) is dominated by a vertex which is not u (v). \( deg(v) ≥ 4 \), and thus (v, u) is not fixed, which contradicts that σ satisfies (P1) by the condition (ii).  \hfill \square

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig4.png}
\caption{Input transformation from σ to σ₃.}
\end{figure}

\textbf{Lemma 4.7} 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) \( \frac{E[C_{RA}(σ)]}{C_{OPT}(σ)} ≤ \frac{E[C_{RA}(σ′)]}{C_{OPT}(σ′)} \).

\textbf{Proof.} In the same way as the proof of Lemma 4.5, we can obtain an input satisfying the property (b) and (P1) in the property (a) in the statement by constructing the two inputs recursively from an input satisfying the conditions (i) and (ii). The degree of any vertex in a newly constructed input does not increase by the definitions of \( f₁ \) and \( f₂ \). Thus, the degree is at most three by (P2) in the condition (ii), which means that this input also satisfies (P2) in (a).  \hfill \square

\textbf{Lemma 4.8} 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) \( \frac{E[C_{RA}(σ)]}{C_{OPT}(σ)} ≤ \frac{E[C_{RA}(σ′)]}{C_{OPT}(σ′)} \).

\textbf{Proof.} We prove this lemma in a similar way to the proofs of the last three lemmas. Suppose that an input σ satisfies the conditions (i) and (ii) in the statement of this lemma. To prove the lemma, we will prove that we can construct an input σ₃ satisfying the following three properties from σ: (1) the number of vertices selected by OPT of degree at most two of the graph in σ₃ is less than that in σ, (2) (P1) and (P2) hold, and (3) \( \frac{E[C_{RA}(σ)]}{C_{OPT}(σ)} ≤ \frac{E[C_{RA}(σ₃)]}{C_{OPT}(σ₃)} \). We can construct an input
satisfying the properties (a), (b) and (c) in the statement recursively by proving the existence of σ₃.

Let v be a vertex of the graph given in σ such that \( \text{deg}(v) \leq 2 \) and OPT selects v. Roughly speaking, we obtain σ₃ by connecting v with a couple of vertices of degree one so that the degree of v is three. Formally, if \( \text{deg}(v) = 2 \), we define \( \sigma_3 = f_3(\sigma, v, \tilde{\sigma}) \), in which \( \tilde{\sigma} = (\{u\}, \varnothing, u) \). If \( \text{deg}(v) = 1 \), we define \( \sigma_3 = f_3(f_3(\sigma, v, \tilde{\sigma}), v, \tilde{\sigma}') \), in which \( \tilde{\sigma}' = (\{u'\}, \varnothing, u') \).

(P2) holds in σ by the condition (ii) in the statement. Also, the input transformation by \( f_3 \) does not change the degrees of the vertices except for v, and the degree of v is three in \( \sigma_3 \). Hence, (P2) also holds in \( \sigma_3 \). By Lemma 4.4,

\[
D_{\text{OPT}}(\sigma_3) = D_{\text{OPT}}(\sigma). \tag{9}
\]

The above facts shows that \( \sigma_3 \) satisfies the property (1) described above. Also, \( C_{\text{OPT}}(\sigma_3) = C_{\text{OPT}}(\sigma) \) by Eq. (9). Since OPT selects neither u nor \( u' \), neither \( (v, u) \) nor \( (v, u') \) is free. That is, (P1) also holds in \( \sigma_3 \) by the condition (ii). Hence, we have shown that \( \sigma_3 \) satisfies the property (2). Moreover, even if a new vertex arrives, the cost of RA does not decrease, and hence

\[
\mathbb{E}[C_{RA}(\sigma_3)] \geq \mathbb{E}[C_{RA}(\sigma)].
\]

σ₃ satisfies the property (3) by this inequality and Eq. (9). \qed

**Lemma 4.9** Suppose that an input \( \sigma \) 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 \( \sigma' \) 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) \( \mathbb{E}[C_{RA}(\sigma)] \leq \mathbb{E}[C_{RA}(\sigma')] \).

**Proof.** We prove this lemma similarly to the proofs of the above lemmas. Suppose that an input \( \sigma \) satisfies the conditions (i) and (ii) in the statement. Let \((v, u)\) be a free edge of the graph in \( \sigma \). Suppose that OPT selects \( v \). To prove the lemma, we will prove that we can construct an input \( \sigma'' \) satisfying the following three properties from \( \sigma \): (1) the number of such free edges \((v, u)\) in \( \sigma'' \) is less than that in \( \sigma \), (2) (P1), (P2) and (P3) hold, and (3) \( \mathbb{E}[C_{RA}(\sigma)] \leq \mathbb{E}[C_{RA}(\sigma'')] \). We can construct an input satisfying the properties (a), (b) and (c) in the statement recursively by proving the existence of \( \sigma'' \).

Define \( \sigma_1 = f_1(\sigma, u) \) and \( \sigma_2 = f_2(\sigma, u) \). Furthermore, we add a few vertices to these inputs in the same way as the proof of Lemma 4.8. Since \( \sigma \) satisfies (P3) by the condition (ii), the degree of \( v \) is three in \( \sigma \). Then, define \( \sigma_1' = f_3(\sigma_1, v, \tilde{\sigma}_1) \), in which \( \tilde{\sigma}_1 = (\{u_1\}, \varnothing, u_1) \). If OPT selects \( u \), define \( \sigma_2' = f_3(\sigma_2, u, \tilde{\sigma}_2) \), in which \( \tilde{\sigma}_2 = (\{u_2\}, \varnothing, u_2) \). Otherwise, define \( \sigma_2' = \sigma_2 \). After the input transformation, the degrees of \( v \) and \( u \) are at most three and the degrees of the other vertices do not change. Also, \( \sigma \) satisfies (P2) by the condition (ii). Thus, \( \sigma_1' (\sigma_2') \) also satisfies (P2). \( D_{\text{OPT}}(\sigma_1') \cup D_{\text{OPT}}(\sigma_2') = D_{\text{OPT}}(\sigma) \) by Lemma 4.3. Both \( D_{\text{OPT}}(\sigma_1') = D_{\text{OPT}}(\sigma_1) \) and \( D_{\text{OPT}}(\sigma_2') = D_{\text{OPT}}(\sigma_2) \) by Lemma 4.4. By the above equalities,

\[
D_{\text{OPT}}(\sigma_1') + D_{\text{OPT}}(\sigma_2') = D_{\text{OPT}}(\sigma). \tag{10}
\]

By this equality, each of the free edges except \((v, u)\) is contained in either \( \sigma_1' \) or \( \sigma_2' \). Whether these edges are fixed does not change. Also, neither \((v, u_1)\) nor \((u, u_2)\) (if any) is free. Thus, \( \sigma_1' (\sigma_2') \) satisfies the above property (1). These facts indicate that \( \sigma_1' (\sigma_2') \) satisfies (P1) because \( \sigma \) satisfies (P1) by the condition (ii). The degree of \( v \) in \( \sigma_1' \) is three. The degree of \( u \) in \( \sigma_2' \) is also three if
OPT selects \( u \) in \( \sigma \). The degrees of the other vertices selected by OPT do not change. Hence, \( \sigma'_1 \) (\( \sigma'_2 \)) satisfies (P3) because \( \sigma \) satisfies (P3) by the condition (ii). By the above argument, \( \sigma'_1 \) (\( \sigma'_2 \)) satisfies the property (2).

In what follows, we show
\[
E[C_{RA}(\sigma)] \leq E[C_{RA}(\sigma'_1)] + E[C_{RA}(\sigma'_2)].
\]

This inequality together with \( C_{OPT}(\sigma'_1) + C_{OPT}(\sigma'_2) = C_{OPT}(\sigma) \) from Eq. (10) shows that the property (3) is satisfied. We evaluate the cost of RA using Lemma 4.2. The input transformations by \( f_1 \) and \( f_2 \) do not affect the vertices except for \( v \), \( u \) and adjacent vertices to either \( v \) or \( u \). The degree of \( v \) does not change. \( u \) becomes \( v_1 \) in \( \sigma'_2 \), for which the expected cost is one. Thus, the expected cost for \( u \) does not decrease. The expected cost for an adjacent vertex to \( u \) does not decrease by the same argument as \( v' \) in the proof of Lemma 4.5. The expected cost for an adjacent vertex to \( u \) does not decrease, either. Now we have shown Eq. (11).

\[\square\]

**Lemma 4.10** Suppose that an input \( \sigma \) satisfies the following conditions: (i) the graph in \( \sigma \) contains at least one good vertex triplet, and (ii) the properties from (P1) to (P4) inclusive hold.

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

**Proof.** We prove this lemma similarly to the proofs of the above lemmas. Suppose that an input \( \sigma \) satisfies the conditions (i) and (ii) in the statement. To prove the lemma, we will prove that we can construct an input \( \sigma'' \) satisfying the following three properties from \( \sigma \): (1) the number of good vertex triplets in the graph given in \( \sigma'' \) is less than that in \( \sigma \). (2) (P1), (P2), (P3) and (P4) hold, and (3) \( \frac{E[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{E[C_{RA}(\sigma'')]}{C_{OPT}(\sigma'')} \). We can construct an input satisfying the properties (a), (b) and (c) in the statement recursively by proving the existence of \( \sigma'' \).

The graph in \( \sigma \) contains a good vertex triplet of \( (u_1, u_2, u_3) \). That is, both \( u_1 \) and \( u_3 \) are adjacent to \( u_2 \), and OPT selects \( u_1 \) and \( u_3 \). Without loss of generality, suppose that \( u_1 \) is revealed before \( u_3 \). If \( u_2 \) is revealed after \( u_1 \), then the edge \( (u_1, u_2) \) is free, which contradicts that \( \sigma \) satisfies (P4) by the condition (ii) because OPT selects \( u_1 \). Thus, \( u_2 \) is revealed before \( u_1 \). Now, define \( \sigma_1 = f_1(\sigma, u_2), \sigma_2 = f_2(\sigma, u_2), \sigma'_2 = f_1(\sigma_2, u_3), \) and \( \sigma'_2 = f_1(\sigma_2, u_1) \) (see Fig. 5). Let \( u \neq u_1, u_3 \) be the vertex adjacent to \( u_2 \). Since \( u \) is dominated by a vertex except for \( u_1 \) and \( u_3 \), either \( (u_2, u) \) or \( (u, u_2) \) is free. Thus,
\[
D_{OPT}(\sigma_1) \cup D_{OPT}(\sigma_2) = D_{OPT}(\sigma)
\]
by Lemma 4.3. Furthermore, \( (u_2, u_1) \) and \( (u_2, u_3) \) are free because OPT selects both \( u_1 \) and \( u_3 \). Hence, \( D_{OPT}(\sigma'_2) \cup D_{OPT}(f_2(\sigma_2, u_3)) = D_{OPT}(\sigma_2) \) and \( D_{OPT}(\sigma'_2) \cup D_{OPT}(f_2(\sigma_2, u_1)) = D_{OPT}(\sigma_2) \) by Lemma 4.3. Since OPT does not select \( u_2 \) in \( \sigma \) by definition, \( D_{OPT}(\sigma'_2) \cap D_{OPT}(\sigma'_2) = \emptyset \). Thus, \( D_{OPT}(f_2(\sigma_2, u_3)) = D_{OPT}(\sigma'_2) \). By these equalities,
\[
D_{OPT}(\sigma'_2) \cup D_{OPT}(\sigma'_2) = D_{OPT}(\sigma_2).
\]
By this equality and Eq. (12),
\[
D_{OPT}(\sigma_1) \cup D_{OPT}(\sigma'_2) \cup D_{OPT}(\sigma'_2) = D_{OPT}(\sigma),
\]
which indicates that the above input transformations do not affect free edges except for \( (u_2, u_1), (u_2, u_3) \) and \( (u, u_2) \) (or \( (u, u_2) \)). Also, the number of good vertex triplets in \( \sigma_1 \) (\( \sigma'_2, \sigma'_2 \)) is less than
that in $\sigma$, and hence these inputs satisfy the property (1). Since $\sigma$ satisfies the properties from (P1) through (P4) by the condition (ii), $\sigma_1$, $\sigma'_2$ and $\sigma''_2$ also satisfy these properties, that is, the property (2).

Finally, we show that at least one input of $\sigma_1$, $\sigma'_2$ and $\sigma''_2$ satisfies the property (3). By Eq. (13),

$$C_{OPT}(\sigma_1) + C_{OPT}(\sigma'_2) + C_{OPT}(\sigma''_2) = C_{OPT}(\sigma).$$

(14)

We evaluate the expected costs for $\sigma_1$, $\sigma'_2$ and $\sigma''_2$ using Lemma 4.2. For $i = 1$ and 3, let $U_i$ be the set of vertices consisting of $u_i$ and all the descendants of $u_i$. Let $U_2$ be the set of all the vertices except for $U_1 \cup U_3$ in $\sigma$. For each $j \in \{1, 2, 3\}$, let $c_j$ be the expected cost of RA for $U_j$ in $\sigma$. By definition,

$$E[C_{RA}(\sigma)] = c_1 + c_2 + c_3.$$  

(15)

Note that the set of all the vertices of the graph given in $\sigma_1$ is $U_2 \setminus \{u_2\}$. The degrees of these vertices except for $u$ do not change before and after the input transformation. Also, the expected cost for $u_2$ in $\sigma$ is one (at most one). Thus,

$$E[C_{RA}(\sigma_1)] \geq c_2 - 2.$$  

(16)

The degrees of all the vertices except for $u_2$ in $\sigma'_2$ are equal to those in $\sigma$, and hence their expected costs do not change. Since $u_2$ is the first revealed vertex in $\sigma_2$, its expected cost does not decrease. Since $U_1$ does not contain $u_2$,

$$E[C_{RA}(\sigma'_2)] \geq c_1 + 1.$$  

(17)

In the same way,

$$E[C_{RA}(\sigma''_2)] \geq c_3 + 1.$$  

(18)

By Eqs. (15), (16), (17) and (18),

$$E[C_{RA}(\sigma_1)] + E[C_{RA}(\sigma'_2)] + E[C_{RA}(\sigma''_2)] \geq c_1 + c_2 + c_3 = E[C_{RA}(\sigma)].$$

The property (3) holds by this inequality and Eq. (14).

---

Figure 5: Construction of $\sigma_1$, $\sigma'_2$ and $\sigma''_2$ from $\sigma$. Note that both $\sigma'_2$ and $\sigma''_2$ contain $u_2$.

---

**Lemma 4.11** Consider the graph in an input satisfying the properties from (P1) to (P6) inclusive. Then, the degree of any vertex is one or three. That is, (P7) holds.

**Proof.** Consider an input satisfying the properties from (P1) to (P6) inclusive. It suffices to prove that the degree of a vertex $v$ in the input is not two because the degree of $v$ is at most three by (P2). Then, we prove it by contradiction, and assume that the degree of $v$ is two.
Let $u$ be a vertex arriving at $v$. The edge $(v, u)$ is not free by (P6). $OPT$ does not select $v$ because the degree of any vertex selected by $OPT$ is three by (P3). Let $v'(\neq u)$ be the other vertex adjacent to $v$. If $OPT$ selects $v'$, then $(v, u)$ is free. Thus, $OPT$ does not select $v'$, which means that $u$ dominates $v$. If $v = v_1$, then the edge $(v, v')$ is free. This edge is not fixed, which contradicts that the input satisfies (P1). If $v \neq v_1$, then the edge $(v', v)$ is free. This edge is not fixed since $\deg(v) = 2$ by definition, which contradicts (P1). By the above argument, we have shown that the degree of $v$ is not two. 

Lemma 4.12 Suppose that an input $\sigma$ 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 $\sigma'$ such that (a) for any free edge $(v, u)$, $\deg(v) = 3$, (b) the properties from (P1) to (P5) inclusive hold, and (c) $\frac{E[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{E[C_{RA}(\sigma')]}{C_{OPT}(\sigma')}$. 

Proof. We prove this lemma similarly to the proofs of the above lemmas. Suppose that an input $\sigma$ satisfies the conditions (i) and (ii) in the statement. Let $(v, u_1)$ be a free edge of the graph in $\sigma$ such that the degree of $v$ is two and there does not exist a descendant $v'$ of $v$ such that $(v', v'')$ is free, in which a vertex $v''$ arrives at $v'$ and the degree of $v'$ is two. To prove the lemma, we will prove that we can construct an input $\sigma''$ satisfying the following three properties from $\sigma$: (1) the number of such free edges $(v, u_1)$ of the graph given in $\sigma''$ is less than that in $\sigma$. (2) the properties from (P1) to (P5) inclusive hold, and (3) $\frac{E[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{E[C_{RA}(\sigma'')]}{C_{OPT}(\sigma'')}$. We can construct an input satisfying the properties (a), (b) and (c) in the statement recursively by proving the existence of $\sigma''$.

Define $\sigma_1 = f_1(\sigma, u_1)$. We fist consider the case in which $\frac{E[C_{RA}(\sigma_1)]}{C_{OPT}(\sigma_1)} \geq \frac{E[C_{RA}(\sigma)]}{C_{OPT}(\sigma)}$. Clearly $\sigma_1$ satisfies the above property (3). $D_{OPT}(\sigma_1)$ can be a subset of $D_{OPT}(\sigma)$ by Lemma 4.3, and thus the number of such edges $(v, u_1)$ in $\sigma_1$ is less than that in $\sigma$ by one, which indicates that the property (1) is satisfied. Furthermore, $\sigma_1$ satisfies the properties from (P1) through (P5) by the condition (ii), and hence satisfies the property (2).

Next we consider the case in which

$$\frac{E[C_{RA}(\sigma_1)]}{C_{OPT}(\sigma_1)} < \frac{E[C_{RA}(\sigma)]}{C_{OPT}(\sigma)}.$$  (19)

Define $\sigma_2 = f_2(\sigma, u_1)$, and let $\sigma_2'$ be a copy of $\sigma_2$. Specifically, define $\sigma_2' = (U', F', R')$, in which $\sigma_2 = (U, F, R)$, $k$ is the number of vertices in $U$, $U = \{u_i \mid i \in [1, k]\}$, $R = \{u_1, \ldots, u_k\}$, $U' = \{u_i' \mid i \in [1, k]\}$, $R' = \{u_1', \ldots, u_k'\}$ and $F' = \{\{u_i', u_j'\} \mid \{u_i, u_j\} \in F\}$. Then, define $\sigma_3 = f_3(\sigma, v, \sigma_2')$. First, we discuss vertices selected by $OPT$ of the graph in $\sigma_3$ to prove that $\sigma_3$ satisfies the properties (1) and (2). Suppose that for vertices $u, u', u'' \in U$, $u$ is adjacent to $u'$, $u'$ is adjacent to $u''$, and $OPT$ selects $u$ in $\sigma$. Then, $OPT$ does not select $u'$ in $\sigma$ because $\sigma$ satisfies (P4) by the condition (ii). Also, the degree of any vertex in $U$ is one or three by the definition of $v$ together with Lemma 4.11. Hence, $OPT$ does not select $u''$ in $\sigma$ because $\sigma$ satisfies (P5) by the condition (ii). The length of a path between two different vertices selected by $OPT$ in $U$ is at least three when dealing with $\sigma$ by these facts. Thus, any vertex in $U$ is dominated by only one vertex in $\sigma$. Moreover, a vertex selected by $OPT$ dominates four vertices by (P3). Thus, the way for $OPT$ to select vertices in $U$ is unique. That is, the vertices selected by $OPT$ in $U$ in $\sigma_3$ are the same as those in $\sigma$ (called Fact (a)). There exists $OPT$ such that $OPT$ selects $u_1$ if and only if $OPT$ selects $u_1'$ by the definition of $U'$, and we use such $OPT$ (Fact (b)). Thus, $v$ is not dominated by only $u_1'$ in $\sigma_3$ (Fact (c)). Let $U''$ be the set of vertices except for $U$ of the graph in $\sigma$, that is, all the vertices in $f_1(\sigma, u_1)$. The set of vertices selected by $OPT$ in $U \cup U''$ for $\sigma_3$ is a dominating set of the graph in $\sigma$ by Fact (c), which
together with Fact (a) and the optimality of OPT for \(\sigma\) shows that the vertices selected by OPT in \(U''\) for \(\sigma_3\) can be the same as those for \(\sigma\) (Fact (d)). Hence, \(\sigma_3\) satisfies the property (1) because the degree of \(v\) in \(\sigma_3\) is three. Also, \(\sigma_3\) satisfies the properties from (P1) through (P5) because \(\sigma\) satisfies the properties by the condition (ii) and \(\sigma_3\) satisfies the property (2).

Finally, we discuss the property (3). Let \(a\) and \(b\) be the expected costs of RA for \(U'' \setminus \{v\}\) and \(U\), respectively, in \(\sigma\). Let \(a'\) and \(b'\) be the costs of OPT for \(U'' \setminus \{v\}\) and \(U\), respectively, in \(\sigma\). Since the degree of \(v\) in \(\sigma\) (\(\sigma_3\)) is two (three), the expected cost for \(v\) in \(\sigma\) (\(\sigma_3\)) is \(1/2\) (one) by Lemma 4.2. OPT does not select \(v\) in \(\sigma\) by (P3) because the degree of \(v\) is two, and hence by Fact (d), OPT does not select \(v\) in \(\sigma_3\), either. By these costs,

\[
\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} = \frac{a + 1/2 + b}{a' + b'}, \quad (20)
\]

and

\[
\frac{\mathbb{E}[C_{RA}(\sigma_3)]}{C_{OPT}(\sigma_3)} = \frac{a + 1 + 2b}{a' + 2b'} \quad (21)
\]

by Fact (b). The expected cost of RA for \(U'' \setminus \{v\}\) in \(\sigma_1\) is equal to that in \(\sigma\) by Lemma 4.2, and thus \(\mathbb{E}[C_{RA}(\sigma_1)] \geq a\). In addition, \(C_{OPT}(\sigma_1) = a'\) by Lemma 4.3 since \((v, u_1)\) is free. These costs for \(\sigma_1\) and Eqs. (19) and (20) show that \(\frac{a'}{a} < \frac{a + 1/2 + b}{a' + b'}\), which is rearranged to \(\frac{a'}{a} < \frac{1/2 + b}{b'}\). This inequality together with Eqs. (20) and (21) yields

\[
\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} < \frac{\mathbb{E}[C_{RA}(\sigma_3)]}{C_{OPT}(\sigma_3)},
\]

which means that \(\sigma_3\) satisfies the property (3).

Now we can show Lemma 4.13 using Lemmas 4.11 and 4.12. 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.

**BlockRoutine**

**Step 1:** \(\ell := 0\) and \(U := \{v_i \mid i = 1, \ldots, n\}\), in which \(n\) is the number of all the vertices of a given graph.

**Step 2:** \(\ell := \ell + 1\). If \(U = \emptyset\), 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 \(U \cap N(v_{i_1}) = \emptyset\), 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 \(U' = \emptyset\), 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 4.13** The number of all the vertices of a given tree is at least four.
Proof. Clearly $OPT$ selects at least one vertex. The degree of a vertex selected by $OPT$ is three by (P3), and thus 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 (Fig. 6): 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 $deg(u_1) = 1$, $deg(u_3) = 3$ and $deg(u_3) = 1$, and a $B_4$-block is a set such that $deg(u_1) = 3$, $deg(u_2) = 3$ and $deg(u_3) = 3$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{blocks.png}
\caption{Blocks}
\end{figure}

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 (Fig. 7). 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 $B_2^{101}$-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.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{classified_blocks.png}
\caption{Classified blocks. $OPT$ selects black vertices.}
\end{figure}

Lemma 4.14 An input can contain at most six kinds of blocks: $B_1^0, B_2, B_3, B_4^{000}, B_4^{100}$ and $B_4^{010}$.
Proof. A $B_1^1$-block does not satisfy (P3). A $B_2^1$-block, a $B_3^1$-block and a $B_4^1$-block do not satisfy (P4). A $B_4^0$-block composes a good vertex triplet, which does not satisfy (P5). □

A $B_1$-block consists of one vertex $v$ of degree one, and (4) in Lemma 4.2 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 4.15 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.

Proof. Suppose that a $B_1$-block consists of a vertex $u_1$, and $u_1$ is adjacent to a vertex $u$. By definition, for a $B_{1,0}$-block, $deg_{u_1}(u) = 3$ holds while for a $B_{1,1}$-block, $deg_{u_1}(u) \leq 2$ holds. By (4) in Lemma 4.2, the expected costs for them are zero and $1/2$, respectively (Fig. 8).

$B_1$-block: Suppose that a $B_1$-block consists of vertices $u_1$, $u_2$ and $u_3$, $deg(u_1) = deg(u_2) = 3$, $deg(u_3) = 1$, and $u_2$ is adjacent to both $u_1$ and $u_3$. The expected cost for $u_1$ ($u_2$) is clearly at most one. Also, by (4) in Lemma 4.2, the expected cost for $u_3$ is at most $1/2$. Thus, the expected cost for a $B_2$-block is at most $5/2$.

$B_3$-block: Suppose that a $B_3$-block consists of vertices $u_1$, $u_2$ and $u_3$, $deg(u_1) = deg(u_3) = 1$ and $deg(u_2) = 3$. Without loss of generality, suppose that $u_1$ is revealed before $u_3$. Since this block does not contain $v_1$ by the assumption of the lemma, $u_2$ is revealed before $u_1$. Hence, $deg_{u_1}(u_2) = 2$ and $deg_{u_3}(u_2) = 3$. By (4-1) and (4-2) in Lemma 4.2, the expected costs for $u_1$ and $u_3$ are $1/2$ and zero, respectively. Also, the expected cost for $u_2$ is clearly at most one. By the above argument, the expected cost for a $B_4$-block is at most $3/2$.

$B_4$-block: A $B_4$-block contains three vertices. The expected cost for each vertex is at most one and thus the expected cost for a $B_4$-block is at most three. □

Figure 8: Costs for blocks without the first revealed vertex. Neither $A$ nor $B$ selects white vertices, both $A$ and $B$ select black vertices, and only the one of $A$ and $B$ selects gray vertices.

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 4.16 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.

Proof. $B_2$-block, $B_4$-block: $B_2$-blocks and $B_4$-blocks consist of three vertices each. The expected cost for any vertex is at most one. Hence, the expected costs for these blocks are at most three each (Fig. 9).

24
**B³-block:** Suppose that a $B³$-block consists of three vertices $u₁, u₂$ and $u₃$, $\text{deg}(u₁) = \text{deg}(u₃) = 1$ and $\text{deg}(u₂) = 3$. Without loss of generality, suppose that $u₁$ is revealed before $u₃$. First, we consider the case in which $u₁ = v₁$. Clearly, the expected costs for $u₁$ and $u₂$ are at most one each. Since $\text{deg}_u(u₂) = 2$ by the definition of BlockRoutine, the expected cost for $u₃$ is $1/2$ by (4-1) in Lemma 4.2. By the above argument, the expected cost for a $B³$-block is at most $5/2$ in this case.

Next, we consider the case in which $u₂ = v₁$. The expected cost for $u₂$ is clearly at most one. The vertex adjacent to $u₂$ which is neither $u₁$ nor $u₃$ is revealed after $u₃$ by the definition of BlockRoutine. Thus, $\text{deg}_u(u₂) = 1$ and $\text{deg}_u(u₃) = 2$. Hence, the expected costs for $u₁$ and $u₃$ are $1/2$ each by (4-1) in Lemma 4.2. By summing up the above costs, the expected cost for a $B³$-block is two in this case. Therefore, the expected cost for a $B³$-block is at most $5/2$.

![Figure 9: Costs for blocks with $v₁$. Both $A$ and $B$ select black vertices and only the one of $A$ and $B$ selects gray vertices.](image)

Let $b₁,ₐ, b₁,₁, b₂, b₃, b₄₀₀₀, b₄₁₀₀$ and $b₄₁₀₁$ denote the numbers of $B₁₀$-blocks, $B₁₁$-blocks, $B₂$-blocks, $B₃$-blocks, $B₄₀₀₀$-blocks, $B₄₁₀₀$-blocks and $B₄₁₀₁$-blocks, respectively. We define $b₄ = b₄₀₀₀ + b₄₁₀₀ + b₄₁₀₁$.

**Lemma 4.17** If the number of all the vertices of a given graph is at least five,

$$b₁,₀ ≤ b₂ + b₄₁₀₀ + b₄₁₀₁$$

and

$$b₁,₁ ≤ b₄₁₀₀.$$

**Proof.** We discuss whether vertices in each block are adjacent to the vertex in a $B₁$-block. If the vertex in a $B₃$-block is adjacent to the vertex in another $B₁$-block, the number of vertices of a given tree is at most four, which contradicts the assumption of this lemma. This discussion is also true with a $B₁$-block. In what follows, we consider $B₂₁₀$-blocks, $B₄₀₀₀$-blocks, $B₄₁₀₀$-blocks and $B₄₁₀₁$-blocks. Since these blocks consist of three vertices each, suppose that they are $u₁, u₂$ and $u₃$, and both $u₁$ and $u₃$ are adjacent to $u₂$.

**$B₂₁₀$-block:** Suppose that $\text{deg}(u₁) = \text{deg}(u₂) = 3$ and $\text{deg}(u₃) = 1$. $\text{OPT}$ selects $u₂$ by definition. Let $u(\neq u₂)$ be a vertex adjacent to $u₁$. $\text{OPT}$ must select $u$ or a vertex adjacent to $u$ to dominate $u$. If $\text{OPT}$ selects $u$, the degree of $u$ is three by (P3). Otherwise, it is at least two because there exists an adjacent vertex to dominate $u$ (i.e., by the assumption of an input, it is exactly three). Since the degree of the vertex in a $B₁$-block is one, $u$ is not contained in a $B₁$-block. Next, let $u′(\neq u₁, u₃)$ be the vertex adjacent to $u₂$. By the definition of BlockRoutine, $u′$ is revealed either before $u₂$ or after both $u₁$ and $u₃$. In the former case, $u′$ is contained in a block of three vertices, which is not a $B₁$-block. In the latter case, $\text{deg}_{u′}(u₂) = 3$. Therefore, $u′$ can be contained only in a $B₁₀$-block.
**$B_{4}^{000}$-block:** OPT does not select any vertex in this block by definition, and hence each vertex is adjacent to a vertex selected by OPT in another block. The degree of the dominating vertex is three by (P3). Both $u_1$ and $u_3$ can be adjacent to a vertex $u''$ which is not selected by OPT. In a similar way to $u$ in the case of a $B_{2}^{000}$-block, the degree of $u''$ is three. Thus, no vertices in a $B_{4}^{000}$-block are adjacent to the vertex in a $B_{1}$-block.

**$B_{4}^{100}$-block:** Suppose that OPT selects $u_1$. The degree of a vertex adjacent to $u_3$ in this block is three similarly to $u_3$ in a $B_{4}^{000}$-block. The degree of a vertex adjacent to $u_2$ is also three similarly to $u$ in the case of a $B_{2}^{010}$-block. We consider the two vertices $u$ and $u'$ which are adjacent to $u_1$, in which $u$ is revealed before $u'$. First, suppose that $u_1$ is revealed before $u_2$. If $u_1 = v_1$, then $deg_u(u_1) = 2$ and $deg_{u'}(u_1) = 3$ by the definition of BlockRoutine. Hence, $u, u'$ can be the vertices in a $B_{1,1}$-block and a $B_{1,0}$-block, respectively, by the definitions of $B_{1,0}$-blocks and $B_{1,1}$-blocks. If $u_1 \neq v_1$, then $u$ is revealed before $u_1$ and $deg_{u'}(u_1) = 3$. Next, suppose that $u_2$ is revealed before $u_1$. Then, $deg_u(u_1) = 2$ and $deg_{u'}(u_1) = 3$. By the above argument, at most one vertex in a $B_{4}^{100}$-block can be adjacent to the vertex in a $B_{1,0}$-block (a $B_{1,1}$-block).

**$B_{4}^{010}$-block:** OPT selects $u_2$ by definition. In a similar way to the above argument, the degree of a vertex adjacent to either $u_1$ or $u_3$ is three. Also, the vertex adjacent to $u_2$ in another block can be contained in a $B_{1,0}$-block.

Therefore, at most one vertex in a $B_{2}^{010}$-block, a $B_{4}^{100}$-block, or a $B_{4}^{010}$-block can be adjacent to the vertex in a $B_{1,0}$-block. That is, Eq. (22) holds. At most one vertex in a $B_{4}^{000}$-block can be adjacent to the vertex in a $B_{1,1}$-block. That is, Eq. (23) holds.

**Lemma 4.18** $b_{1,0} + b_{1,1} + b_{3} = b_{2} + 3b_{4} + 2$.

**Proof.** Let $i_2$ denote the number of edges each of which is an edge between a vertex in a $B_{2}$-block and a vertex in either another $B_{2}$-block or a $B_{4}$-block. Let $o_2$ denote the number of edges each of which is an edge between a vertex in a $B_{2}$-block and a vertex in either a $B_{1}$-block or a $B_{3}$-block. Similarly, let $i_4$ denote the number of edges each of which is an edge between a vertex in a $B_{4}$-block and a vertex in either another $B_{4}$-block or a $B_{2}$-block. Let $o_4$ denote the number of edges each of which is an edge between a vertex in a $B_{4}$-block and a vertex in either a $B_{1}$-block or a $B_{2}$-block. By definition,

$$b_{1,0} + b_{1,1} + b_{3} = o_{2} + o_{4}.$$  \(24\)

Since there exist three edges each of which is between a vertex in a $B_{2}$-block and one in another block,

$$3b_{2} = i_{2} + o_{2}.$$  \(25\)

Since there exist five edges each of which is between a vertex in a $B_{4}$-block and one in another block,

$$5b_{4} = i_{4} + o_{4}.$$  \(26\)

Let us call a vertex of degree at least two an internal vertex. For a tree, $x = y + 1$ holds, in which $x$ is the number of internal vertices and $y$ is the number of edges between internal vertices. If we regard a block as one vertex, then the number of internal vertices is $b_{2} + b_{4}$ and the number of edges between internal vertices is $(i_{2} + i_{4})/2$. Thus,

$$b_{2} + b_{4} = (i_{2} + i_{4})/2 + 1.$$  \(27\)

By summing up Eqs. (25) and (26),

$$3b_{2} + 5b_{4} = i_{2} + o_{2} + i_{4} + o_{4} = b_{1,0} + b_{1,1} + b_{3} + 2b_{2} + 2b_{4} - 2.$$  \(\text{by Eqs. (24) and (27)}\)
We complete the proof by rearranging this equality. □

**Theorem 4.19** The competitive ratio of RA is at most $5/2$.

**Proof.** First, we consider an input $\sigma$ 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 4.14. 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 4.16. Let $v$ denote the vertex in $C_1$, and let $u$ denote the vertex adjacent to $v$ in $C_3$. $\text{deg}_v(u) = 3$ 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 4.15. By the above argument, $\mathbb{E}[C_{RA}(\sigma)] = 5/2$. On the other hand, $OPT$ clearly selects at least one vertex, that is, $C_{OPT}(\sigma) \geq 1$. 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 4.15 and 4.16. Also, $v_1$ does not affect the expected cost for $B_1$-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 $b'_2 + b'_3 \leq 1$. Then, using Lemma 4.15, we have

$$\mathbb{E}[C_{RA}(\sigma)] \leq b_{1,1}/2 + 5b_2/2 + 3b_3/2 + 3b_4 + b'_2/2 + b'_3 \leq b_{1,1}/2 + 5b_2/2 + 3b_3/2 + 3b_4 + 1.$$  

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

$$\frac{\mathbb{E}[C_{RA}(\sigma)]}{C_{OPT}(\sigma)} \leq \frac{b_{1,1}/2 + 5b_2/2 + 3b_3/2 + 3b_4 + 1}{b_2 + b_3 + b_4^{100} + b_4^{010}}$$

$$= \frac{-3b_{1,0}/2 - b_{1,1} + 4b_2 + 15b_4/2 + 4}{-b_{1,0} - b_{1,1} + 2b_2 + 3b_3 + b_4^{100} + b_4^{010} + 2}$$

(by the substitution for $b_3$ by Lemma 4.18)

$$\leq \frac{-3b_{1,0}/2 + 4b_2 + 15b_4/2 - b_4^{100} + 4}{-b_{1,0} + 2b_2 + 3b_4 + b_4^{100} + 2}$$

(by the substitution for $b_{1,1}$ by Eq. (23))

$$\leq \frac{-5b_{1,0}/2 + 5b_2 + 15b_4/2 + b_4^{100} + 4}{-b_{1,0} + 2b_2 + 3b_4 + b_4^{100} + 2}$$

(by the substitution for $b_4^{100}$ by Eq. (22))

$$= \frac{5}{2} \cdot \frac{-b_{1,0} + 2b_2 + 3b_4 + 2b_4^{100}/3 + 8/5}{-b_{1,0} + 2b_2 + 3b_4 + b_4^{100} + 2} < \frac{5}{2}.$$  

□

Our analysis of RA is exact by the following theorem.

**Theorem 4.20** The competitive ratio of RA is at least $5/2$.

**Proof.** Consider the following input $\sigma$. A vertex $v_1$ is first revealed, and the second vertex $v_2$ arrives at $v_1$. The third vertex $v_3$ and the fourth vertex $v_4$ arrive at $v_2$. Then, $A$ selects $v_1$ (Case 1) and $v_2$ (Case 2.2.2). $B$ selects $v_1$ (Case 1), $v_2$ (Case 2.1) and $v_3$ (Case 2.2.1). $OPT$ selects $v_2$. Therefore, $\mathbb{E}[C_{RA}(\sigma)]/C_{OPT}(\sigma) = 5/2$. □
5 Randomized Lower Bound

Lemma 5.1 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 \( p_u + p_v \geq p'_u + 1 - p'_u = 1 \).

\[ \square \]

Theorem 5.2 The competitive ratio of any randomized online algorithm is at least 4/3.

Proof. Consider a randomized online algorithm \( RON \) for the following input \( \sigma \). Let \( m \) be any positive integer. We sketch an adversary constructing \( \sigma \). 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 \( i = 1, 2, \ldots, 2m \), the \( i \)-th vertex \( v_i \) arrives at \( v_{i-1} \). For each \( j = 1, 2, \ldots, 2m \), let \( p_j \) be the probability that \( v_j \in D_{RON}(v_{2m}) \). Next, for each \( \ell = 1, 2, \ldots, m \), 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 \( u_{\ell_1} \) arrives at \( v_{\ell_1} \). By Lemma 5.1, 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 \( p_{\ell_1} \leq 2/3 \) by the condition of Case 2, \( p_{\ell_2} \geq 1/3 \). Hence, the expected cost for \( v_{\ell_1}, v_{\ell_2} \) and \( u_{\ell_1} \) is at least \( 1 + 1/3 = 4/3 \).

Let \( x \) be the number of such \( \ell \) with applying Case 1. Thus, the number of such \( \ell \) 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 \( \ell \) with applying Case 1, \( OFF \) selects \( v_{2\ell-1} \) and for such \( \ell \) with applying Case 2, selects \( v_{\ell_1} \). 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, \( \mathbb{E}[C_{RON}(\sigma)]/C_{OPT}(\sigma) \geq 4/3 \).

\[ \square \]

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, Cambridge, 1998.

[2] J. Boyar, S. J. Eidenbenz, L. M. Favrholdt, M. Kotrbčík and K. S. Larsen, “Online dominating set,” In Proc. of the 15th Scandinavian Symposium and Workshops on Algorithm Theory, pp. 21:1–21:15, 2016.

[3] D.-Z. Du and P.-J. Wan, “Connected Dominating Set: Theory and Applications,” Springer, New York, 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, pp. 245–256, 2004.

[6] F.V. Fomin, F. Grandoni, and D. Kratsch, “Some new techniques in design and analysis of exact (exponential) algorithms,” Bulletin of the EATCS, pp. 47–77, 2005.

[7] M. Henning and A. Yao, “Total Domination in Graphs,” Springer, New York, 2013.

[8] W. Goddard and M. A. Henning, “Independent domination in graphs: A survey and recent results,” Discrete Mathematics, Vol. 313, No. 7, pp. 839–854, 2013.

[9] F. Grandoni, “Independent domination in graphs: A survey and recent results,” Journal of Discrete Algorithms, Vol. 4, No. 2, pp. 209–214, 2006.

[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, pp. 41–54, 2011.

[11] G. H. King and W. G. Tzeng, “On-line algorithms for the dominating set problem,” Information Processing Letters, Vol. 61, pp. 11–14, 1997.

[12] F. Plastria, “Static competitive facility location: An overview of optimisation approaches,” European Journal of Operational Research, Vol. 129, No. 3, pp. 461–470, 2001.

[13] J. M. M. van Rooij and H. L. Bodlaender, “Exact algorithms for dominating set,” Discrete Applied Mathematics, Vol. 159, No. 17, pp. 2147–2164, 2011.
[14] I. Schiermeyer, “Exact algorithms for dominating set,” *Discrete Applied Mathematics*, Vol. 156, No. 17, pp. 3291–3297, 2008.

[15] D. D. Sleator and R. E. Tarjan, “Amortized efficiency of list update and paging rules,” *Communications of the ACM*, Vol. 28, No. 2, pp. 202–208, 1985.

[16] J. Y. Yu and P. H. J. Chong, “A survey of clustering schemes for mobile ad hoc networks,” *IEEE Communications Surveys and Tutorials*, Vol. 7, No. 1, pp. 32–48, 2005.