Hardness and Inapproximability Results for Minimum Verification Set and Minimum Path Decision Tree Problems

Uraz Cengiz Türker Hüsnü Yenigün

Abstract

Minimization of decision trees is a well studied problem. In this work, we introduce two new problems related to minimization of decision trees. The problems are called minimum verification set (MinVS) and minimum path decision tree (MinPathDT) problems. Decision tree problems ask the question "What is the unknown given object?". MinVS problem on the other hand asks the question "Is the unknown object z?", for a given object z. Hence it is not an identification, but rather a verification problem. MinPathDT problem aims to construct a decision tree where only the cost of the root-to-leaf path corresponding to a given object is minimized, whereas decision tree problems in general try to minimize the overall cost of decision trees considering all the objects. Therefore, MinVS and MinPathDT are seemingly easier problems. However, in this work we prove that MinVS and MinPathDT problems are both NP-complete and cannot be approximated within a factor in $o(\lg n)$ unless P = NP.

1 Introduction

Decision trees have been studied extensively (e.g. see [13] for an early survey) and have many practical applications in a wide range of fields, such as databases, switching theory, pattern recognition, taxonomy, medical diagnosis, etc. In this work, we are interested in the minimization of decision trees for object (or entity) identification, where binary tests (or queries) are used to classify the objects. The topic is well explored and there is quite a large number of works on the minimization of decision trees.

In binary identification procedure, there are n objects and m binary tests where the response of an object to a test is either 0 or 1. The purpose of the problem is to design a deterministic procedure where a set of tests are applied to an unknown object to identify the object. Such a procedure can be described by using a binary decision tree where the leaves correspond to the objects and the other nodes correspond to the tests.

As for the hardness results on these problems, Hyafil and Rivest showed that it is NP-complete to decide if a decision tree with a certain expected cost exists when tests have some associated costs [8]. They also showed, in the same work, that the problem remains NP-complete if a probability of occurrence is assigned to each object, if all tests have the same cost, and if worst case is considered instead of expected cost.

There are some works also on the approximation of optimal binary decision trees. Laber and Nogueria proved that there is no $o(\lg n)$ approximation to minimizing the height of the decision tree unless P= NP [10]. In [3], Chakaravarthy et al. considered both the binary and K–ary tests, where the occurrence probability of objects may or may not be uniform. They call the problem $\mathcal{K}$–$\mathcal{DT}$ when K–ary tests are considered and when each object has a given certain probability of occurrence. If the tests are binary, then the problem is called 2–$\mathcal{DT}$. Similarly, the problem is called $\mathcal{K}$–$\mathcal{UDT}$ and 2–$\mathcal{UDT}$, when each object has the same probability, but the tests are K–ary and binary, respectively. They provide both approximation algorithms and inapproximability results for these four types of problems separately. The cost of a decision tree is the expected cost over all the the objects. Adler and Heeringa present an $(\ln n+1)$ approximation for 2–$\mathcal{UDT}$ and 2–$\mathcal{DT}$ problems in [1].

All works in the literature are related to the minimization of decision trees with respect to the expected cost of identification or the worst case cost of the identification. In this work, we are interested in a slightly different but a related problem. Instead of identifying an unknown object, we want to verify if the unknown object is a certain object z or not. This can be accomplished by using a subset of tests which we call as a *verification set for the object z*. The minimization problem in this context is to find a minimum verification set. We came across this problem as we were working on a related minimization problem within the context of finite state machine based testing using adaptive distinguishing sequences [11]. To the best of our knowledge, the minimum verification

set problem has not been studied before. Since the minimum verification set problem focuses on a given object, it seems to be an easier problem than the decision tree related problems where all the objects are considered. However, we show that the decision version of the minimum verification set problem is NP-complete. We also show that, the problem cannot be approximated within a factor in $o(\lg n)$ unless P = NP. In addition, we study a directly related problem of finding a decision tree where the root-to-leaf path corresponding a certain given object is minimized. We call this problem MinPathDT. This problem also turns out to be NP-complete which cannot be approximated again within a factor in $o(\lg n)$ provided that P $\neq$ NP.

The paper is organized as follows. Section 2 gives formal definitions of binary identification problem, binary decision trees used for the identification, and also gives some notation we use throughout the paper. In Section 3, MinVS problem is introduced formally. The hardness and inapproximability results for MinVS problem are also given in Section 3. Section 4 presents the formal definition of MinPathDT problem, and gives the hardness and the inapproximability results for this problem. Finally, Section 5 concludes the paper summarizing our results and giving some pointers for the extension of our work.

2 Preliminaries

Suppose that we are given a rooted tree $\mathcal{A}$ where the vertices and the edges are labeled. The term *internal vertex* is used to refer to a node which is not a leaf. For two vertices p and q in $\mathcal{A}$, we say p *is under* q, if p is a vertex in the subtree rooted at vertex q. A vertex is by definition under itself. For a child p' of p, if the label of the edge from p to p' is l, then we call p' as the *l–successor of p*. In this work, we will always have distinct labels for the edges emanating from an internal node, hence l–successor of a node will always be unique.

2.1 Binary Identification Problem

Let $Z = \{z_1, z_2, \ldots, z_n\}$ be a finite set of distinct objects and $T = \{t_1, t_2, \ldots, t_m\}$ be a finite set of tests where each test $t \in T$ is a function $t : Z \to \{0, 1\}$. Intuitively, when a test t is applied to an object z, the object z produces the response $t(z)$, i.e. either a 0 or a 1 is obtained as an answer.

The set of objects Z and the set of tests T can also be presented as a table $D[T, Z]$ (which we will call as a *decision table*) with m rows and n columns where the rows are indexed by the tests and the columns are indexed by the objects. An element $D[t, z]$ is set to the value $t(z)$. Table 1 is an example of such a decision table where there are 4 objects and 3 tests. A row corresponds to a test t and it gives the vector of responses of the objects to t. Similarly, a column corresponds to an object z and it gives the vector of responses of z to the tests. For a test t and an object z, we will use the notation $D[t, .]$ and $D[., z]$ to refer to the row of $D[T, Z]$ corresponding to the test t and the column of $D[T, Z]$ corresponding to the object z, respectively.

D	z_1	z_2	z_3	z_4
t_1	0	1	1	0
t_2	1	0	1	0
t_3	1	0	1	1

Table 1: An example decision table

For a decision table $D[T, Z]$, if for all objects $z \in Z$, $D[., z]$ is unique, then $D[T, Z]$ is called a *unique response* decision table. Suppose that we are given a unique response decision table $D[T, Z]$ and an unknown object from Z, and we are asked to identify this object. One can apply all the tests in T and the results of the tests will be corresponding to a unique column of the table, identifying the unknown object. Throughout the paper, we will only consider unique response decision tables, as otherwise such an identification is not possible.

Let us call a row/column as *all–0* (resp. *all–1*) if every element in the row/column is 0 (resp. 1). Note that if for a test t, $D[t, .]$ is an all–0 or an all–1 row, this means t does not distinguish between any objects. We call such tests as *useless* since they provide no information for the identification of the unknown object. If a decision table $D[T, Z]$ has useless tests, one can eliminate such tests in polynomial time, by performing a single pass over $D[T, Z]$. It is possible, on the other hand, to have an all–0 (resp. an all–1) column in $D[T, Z]$, as an object z may always have 0 (resp. 1) as the response to all the tests. Two different tests t and t' for which $D[t, .]$ and $D[t', .]$ are the same are called *duplicate tests*. The information provided by t and t' are the same. If a decision table $D[T, Z]$ has duplicate tests, one can eliminate such tests in polynomial

time, by checking the equality of every pair of rows of $D[T, Z]$. A decision table is said to be *reduced* if it has no useless and duplicate tests. We will only consider reduced decision tables, unless stated otherwise.

2.2 Binary Decision Trees

Identifying an unknown object of Z by using tests in T can also be performed adaptively. In this case the procedure to be applied can be described in the form of a binary decision tree $\mathcal{A}$ having the following properties.

Definition 1. *A* decision tree *for a given decision table $D[T, Z]$ where $Z = \{z_1, z_2, \ldots, z_n\}$ and $T = \{t_1, t_2, \ldots, t_m\}$ is a rooted tree $\mathcal{A}$ with n leaves such that:*

(1) Each leaf of $\mathcal{A}$ is labeled by a distinct object $z \in Z$.

(2) Each internal node of $\mathcal{A}$ is labeled by a test $t \in T$.

(3) Each internal node has two outgoing edges, one with label 0 and the other with label 1.

(4) Consider a path from the root to a leaf node p labeled by an object z. Let q be an internal node on this path and t be the test labeling the node q. If p is under the 0–successor of q then $t(z) = 0$, and if p is under the 1–successor of q then $t(z) = 1$.

Figure 1 and Figure 2 present two different decision trees for the decision table given in Table 1. The identification procedure based on a given decision tree $\mathcal{A}$ proceeds as follows: If r is the root node of $\mathcal{A}$, we start by applying the test labeling r. If the outcome is 0, then the subtree rooted at the 0–successor of r is considered, otherwise (when the outcome is 1) the subtree rooted at the 1–successor of r is considered. The procedure is repeated recursively for the root of each subtree visited, until a leaf is reached. When a leaf node p is reached, the object labeling p gives the unknown object.

Note that it is always possible to find such a decision tree thanks to the assumption that $D[T, Z]$ is a unique response decision table (see Observation 1 in [5]).

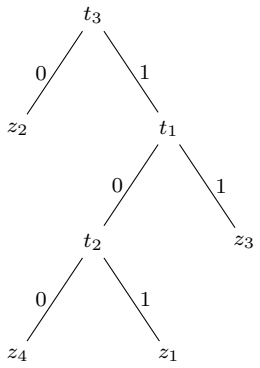

Figure 1: A decision tree for the decision table of Table 1

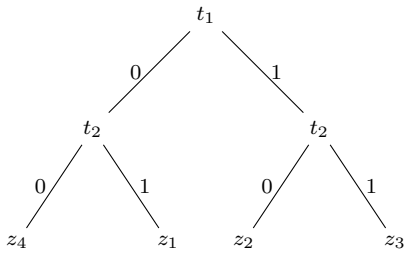

Figure 2: Another decision tree for the decision table of Table 1

3 Minimum Verification Set Problem

Instead of trying to identify an unknown object from Z by using the tests in T, another interesting question can be the following. Given a certain object $z \in Z$, find a set of tests that would check if the unknown object is z or not. Of course, since we assume that $D[., z]$ is unique, this check can be performed by applying all the tests in T. However, we may be able to perform this check by using a subset of tests in T.

Definition 2. *Let $D[T, Z]$ be a decision table and $z \in Z$ be an object. A subset of tests $T' \subseteq T$ is said to be a* verification set for z*, if for any $z' \in Z \setminus \{z\}$, $\exists t \in T'$ such that $t(z) \neq t(z')$. A verification set T' for z is called a* minimum verification set for z *if there does not exist a verification set T'' for z such that $|T''| < |T'|$.*

In other words, any object can be distinguished from z by using some test in T'. For the trivial case where $Z = \{z\}$, the minimum verification set for z is simply the empty set. Note that T itself is always a verification set for any object. One may want to minimize the effort for such a verification hence a verification set with minimal cardinality is desirable. Definition 3 states the problem formally.

Definition 3. MinVS problem: *Given a decision table $D[T, Z]$ and an object $z \in Z$, find a minimum verification set for z.*

We will show the hardness of MinVS using a reduction from the set covering problem. The set covering problem is defined by a tuple (U, C) where $U = \{u_1, u_2, \ldots, u_p\}$ is a (universal) set of items, and $C = \{c_1, c_2, \ldots, c_q\}$ is a collection of non–empty subsets of U (i.e. $\forall c \in C, c \subseteq U$), with the property that

$$\bigcup_{c \in C} c = U$$

A subset $C' \in C$ is said to be a *cover for* U if

$$\bigcup_{c \in C'} c = U$$

The set covering problem (SCP) is to find a minimum cardinality cover C' for U, i.e. a cover where $|C'|$ is minimized.

It is well known that SCP is NP-complete [9, 6]. We will consider only nontrivial SCP instances with the following property: For each $c_i \in C$, $c_i \neq U$. If there exists a subset c_i such that $c_i = U$, then for such instances, SCP has the trivial solution consisting of c_i only. The hardness of SCP is therefore not due to such trivial instances, but due to nontrivial instances which are defined above. Also, we assume that for any two different subsets c_i and c_j in C, we have $c_i \neq c_j$, since such repeated occurrences of subsets can be detected and eliminated in polynomial time by pairwise comparison of the subsets.

3.1 Hardness of Minimum Verification Set Problem

In this section, we will show that the decision version of MinVS is NP-complete. We will use a reduction from SCP which we explain now. Let (U, C) be an instance of SCP. We form the following decision table $D[T, Z]$ from the instance (U, C) of SCP. We call this translation as the mapping β.

- $T = \{t_c | c \in C\}$
- $Z = Z_U \cup \{z^\star\}$, where $Z_U = \{z_u | u \in U\}$ and $z^\star$ is an additional object not in Z_U.
- $\forall t_c \in T, z_u \in Z_U$

$$t_c(z_u) = \begin{cases} 1 & \text{if } u \in c \\ 0 & \text{otherwise} \end{cases}$$

- $\forall t_c \in T$, $t_c(z^\star) = 0$

It is easy to see that, if there are p elements in U and q subsets in C, then there are $p+1$ objects in Z and q tests in T. Note that, since a subset $c \in C$ is non–empty, the row $D[t_c, .]$ is not all–0. Also since we consider nontrivial instances of SCP, for a subset $c \in C$, we also have $c \neq U$, therefore the row $D[t_c, .]$ is not all–1. The assumption that for any two different subsets c_i and c_j we have $c_i \neq c_j$, makes sure that there are no duplicate tests in T. For an object $z_u \in Z_U$, the column $D[., z_u]$ can be an all–1 column, when u happens to belong to every subset $c \in C$. However, $D[., z_u]$ cannot be an all–0 column, since u must belong to at least one subset $c \in C$. The column $D[., z^\star]$ on the other hand is an all–0 column, and it is therefore the only all–0 column in $D[T, Z]$.

We will use the notation $\beta(U, C)$ to denote the decision table $D[T, Z]$ generated by the mapping β from an instance (U, C) of SCP. For a given subset C' of C, we use the notation $\alpha(C')$ to denote the set $\{t_c | c \in C'\}$ and for a given subset T' of the tests of $\beta(U, C)$, we use the notation $\alpha^{-1}(T')$ to denote the set $\{c | t_c \in T'\}$.

Lemma 1. *Let (U, C) be an instance of* SCP *and $D[T, Z] = \beta(U, C)$. Given a cover C' for U, $\alpha(C')$ is a verification set for $z^\star$ and $|\alpha(C')| = |C'|$. Also given a verification set T' for $z^\star$, $\alpha^{-1}(T')$ is a cover for U and $|\alpha^{-1}(T')| = |T'|$.*

Proof. Let C' be a cover for U and consider the set of tests $T' = \alpha(C')$. We will now show that T' is a verification set for $z^\star$. We have $Z \setminus \{z^\star\} = Z_U$. Therefore we need to show that for any $z_u \in Z_U$, there exists a test $t_c \in T'$ such that $t_c(z_u) \neq t_c(z^\star)$. Since C' is a cover for U, for any $u \in U$, there has to be a set $c \in C'$ such that $u \in c$. Then, for the test t_c, we have $t_c(z_u) = 1 \neq t_c(z^\star) = 0$.

Now let T' be a verification set for $z^\star$ and consider the set $C' = \alpha^{-1}(T')$. We will now show that C' is a cover for U, by proving for any $u \in U$, there exists a $c \in C'$ such that $u \in c$. Since $t_c(z^\star) = 0$ for any test t_c, for T' to be a verification set for $z^\star$, for any $z_u \in Z_U$, there must be a test $t_c \in T'$ such that $t_c(z_u) = 1$, implying $u \in c$. That is, for any $u \in U$, there must be a set $c \in C'$ such that $u \in c$, which shows that C' is a cover for U.

In both cases, it is easy to see that $|\alpha(C')| = |C'|$ and $|\alpha^{-1}(T')| = |T'|$ due to the one–to–one correspondence of the tests in $D[T, Z]$ and the subsets in (U, C) by the mapping β. □

We can now show that the decision version MinVS is NP-complete.

Theorem 1. *The decision version of* MinVS *problem is* NP-complete.

Proof. Let K be a constant, $D[T, Z]$ be a decision table, and z be an object in Z. The decision version of MinVS problem asks if there exists a verification set T' for z such that $|T'| \leq K$. Given a set T' such that $|T'| \leq K$, one can check if T' is a verification set for z, by comparing $t(z)$ and $t(z')$ for every $z' \in Z \setminus \{z\}$ and for every $t \in T'$, in polynomial time. Hence, the problem is in NP.

Let (U, C) be an instance of SCP and let $D[T, Z] = \beta(U, C)$. Suppose that it is possible to decide if there is a verification set T' for $z^\star$ such that $|T'| \leq K$ in polynomial time. Then, we can also check if there exists a set cover C' such that $|C'| \leq K$ using the same algorithm, based on Lemma 1. However, we know that SCP is NP-complete. □

3.2 Inapproximability of Minimum Verification Set Problem

There are inapproximability results in the literature for the minimization version of SCP. In [12, 4], it was shown that SCP cannot be approximated within a factor in $o(\lg n)$ unless NP has quasipolynomial time algorithms. It was also shown that SCP does not admit an $o(\lg n)$ approximation under the weaker assumption that $\mathsf{P} \neq \mathsf{NP}$ [14, 2].

Due to the construction of the mapping β, it is also possible to deduce such inapproximability results for MinVS problem. We will first show the relation between the optimal solution of an SCP instance (U, C) and the optimal solution of the corresponding VSP instance $\beta(U, C)$.

Lemma 2. *Given an* SCP *instance* (U, C) *let* $D[T, Z] = \beta(U, C)$. *Let* OPT_{sc} *be the optimal solution of* (U, C) *and* OPT_{vs} *be the optimal solution for the* VSP *instance of finding a verification set for* $z^\star$ *in* $D[T, Z]$. *Then* $OPT_{sc} = OPT_{vs}$.

Proof. Let C' and T' be a cover for U and a verification set for $z^\star$ achieving OPT_{sc} and OPT_{vs}, respectively. Suppose that $OPT_{sc} < OPT_{vs}$, meaning that $|C'| < |T'|$. Using Lemma 1, $T'' = \alpha(C')$ is also a verification set for $z^\star$ and $|T''| = |C'|$. However, this means $|T''| < |T'|$ which is not possible since we know that T' is an optimal solution. Conversely, suppose that $OPT_{sc} > OPT_{vs}$, meaning that $|C'| > |T'|$. Using Lemma 1, $C'' = \alpha^{-1}(T')$ is also a cover for U and $|C''| = |T'|$. However, this means $|C'| > |C''|$ which is not possible since we know that C' is an optimal solution. □

Theorem 2. MinVS *does not admit an* $o(\lg n)$ *approximation algorithm unless* $\mathsf{P} = \mathsf{NP}$.

Proof. Suppose that $\mathsf{P} \neq \mathsf{NP}$ and there exists a polynomial algorithm $\mathcal{P}$ which gives an $o(\lg n)$ approximation for MinVS. In this case, for a given SCP instance (U, C), one can consider $D[T, Z] = \beta(U, C)$, and using $\mathcal{P}$, get a solution T' which is an $o(\lg n)$ approximation for the verification set for $z^\star$ for $D[T, Z]$. In this case, Lemma 1 and Lemma 2 together imply that $\alpha^{-1}(T')$ is also an $o(\lg n)$ approximation for the SCP instance (U, C), which we know to be impossible when $\mathsf{P} \neq \mathsf{NP}$. □

4 Minimum Path Decision Tree Problem

As noted before, there always exists a decision tree for a given decision table $D[T, Z]$ based on the assumption that $D[T, Z]$ is a unique response decision table. However, for a given decision table, there can be more than one decision tree. For example, Figure 1 and Figure 2 are two different decision trees for the decision table given in Table 1. Since the identification procedure for the unknown object is directly based on the decision tree used, the cost of the procedure depends on the decision tree. One may want to minimize this effort by using an appropriate decision tree. There can be different metrics that can be used to measure the effort. Given a decision tree $\mathcal{A}$ and an object $z \in Z$, let $d_{\mathcal{A}}(z)$ be the depth of the leaf node labeled by z in $\mathcal{A}$. For the decision tree in Figure 1, we have $d_{\mathcal{A}}(z_1) = 3$, $d_{\mathcal{A}}(z_2) = 1$, $d_{\mathcal{A}}(z_3) = 2$, and $d_{\mathcal{A}}(z_4) = 3$.

One problem can be to minimize the expected number of tests to be applied, which corresponds to minimizing the sum $\sum_{z \in Z} d_{\mathcal{A}}(z)$ assuming each object is equiprobable. We will call this problem as MinDT problem. Another problem can be to minimize the depth of the decision tree $\mathcal{A}$ in order to minimize the worst case behaviour of the identification procedure based on $\mathcal{A}$. We will call this problem as MinHeightDT problem. It is known that decision versions of the problems MinDT and MinHeightDT are NP-complete (for MinDT problem see [7] and Decision Tree problem (MS15) in [6], for MinHeightDT problem see the concluding remarks in [7]).

In this section, we will consider another metric for the minimization of decision trees. To motivate the problem consider the following scenario. For a given decision table $D[T, Z]$, suppose that the objects are diagnoses in a medi-

cal emergency room where some binary tests are applied to reach a diagnosis. The tests all take the same amount of time, however one of the diagnosis is more important than the others, since it requires a much more urgent action to be taken. In such a case, the situation can be modeled as a binary identification problem, where one would like to find a decision tree whose root–to–leaf path corresponding to this urgent diagnosis is minimized. Definition 4 states the problem formally.

Definition 4. MinPathDT *problem: Given a decision table* $D[T, Z]$ *and an object* $z \in Z$*, find a decision tree* $\mathcal{A}$ *such that* $d_{\mathcal{A}}(z)$ *is minimized.*

In the following sections, we will show the hardness and inapproximability of the MinPathDT problem.

4.1 Hardness of Minimum Path Decision Tree Problem

Consider a decision tree $\mathcal{A}$ and a leaf vertex p labeled by an object z in $\mathcal{A}$. We use the notation $\mathcal{A}|_z$ to denote the set of internal vertex labels on the path from the root of $\mathcal{A}$ to p. For example, for the decision tree $\mathcal{A}$ given in Figure 1, $\mathcal{A}|_{z_2} = \{t_3\}$, $\mathcal{A}|_{z_3} = \{t_1, t_3\}$ and $\mathcal{A}|_{z_1} = \mathcal{A}|_{z_4} = \{t_3, t_1, t_2\}$.

We will show the relation between solving MinVS for an object z and solving MinPathDT for the same object z. Basically, the idea is to show that given a minimum verification set T' for an object z, it is always possible to build a decision tree $\mathcal{A}$ such that $\mathcal{A}|_z = T'$.

If $D[T, Z]$ is a decision table, one can consider a subset Z' of objects, and form the table $D[T, Z']$ which would still be a decision table. If $D[T, Z]$ is a unique response decision table, then so is $D[T, Z']$. However, even though $D[T, Z]$ is reduced, $D[T, Z']$ may not be reduced. Some tests in $D[T, Z']$ may be duplicate and/or useless. As explained in Section 2.1 though, it is always possible to get a reduced decision table from $D[T, Z']$ by eliminating duplicate and useless tests, in polynomial time.

For a value $x \in \{0, 1\}$, let $\bar{x}$ denote the negation of the value x, i.e. $\bar{x} = 1 - x$. We will first introduce a textual notation to describe trees using the following grammar. For any object $z \in Z$, z is a tree. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two trees, and t be a test. In this case, $\langle A_1 \overset{x}{\leftarrow} t \overset{\bar{x}}{\rightarrow} A_2 \rangle$ is a tree, where $x \in \{0, 1\}$. Using this notation, the decision trees in Figure 1 and Figure 2 are given as $\langle z_2 \overset{0}{\leftarrow} t_3 \overset{1}{\rightarrow} \langle \langle z_4 \overset{0}{\leftarrow} t_2 \overset{1}{\rightarrow} z_1 \rangle \overset{0}{\leftarrow} t_1 \overset{1}{\rightarrow} z_3 \rangle \rangle$ and $\langle \langle z_4 \overset{0}{\leftarrow} t_2 \overset{1}{\rightarrow} z_1 \rangle \overset{0}{\leftarrow} t_1 \overset{1}{\rightarrow} \langle z_2 \overset{0}{\leftarrow} t_2 \overset{1}{\rightarrow} z_3 \rangle \rangle$,

respectively. Note that a tree given in this notation is not necessarily a decision tree, however all decision trees can be described using this notation.

Lemma 3. *Let $D[T, Z]$ be a decision table and $Z_1 \subset Z$ and $Z_2 \subset Z$ be two subsets of objects such that $Z_1 \cap Z_2 = \emptyset$. Let $\mathcal{A}_1$ be a decision tree for $D[T, Z_1]$ and $\mathcal{A}_2$ be a decision tree for $D[T, Z_2]$. Furthermore, let $t \in T$ be a test such that $\forall z_1, z_1' \in Z_1, \forall z_2, z_2' \in Z_2, t(z_1) = t(z_1') = x \neq \bar{x} = t(z_2) = t(z_2')$. In this case, $\mathcal{A} = \langle \mathcal{A}_1 \overset{x}{\leftarrow} t \overset{\bar{x}}{\rightarrow} \mathcal{A}_2 \rangle$ is a decision tree for $D[T, Z_1 \cup Z_2]$.*

Proof. Since $\forall z_1, z_1' \in Z_1$, $t(z_1) = t(z_1')$, i.e. t cannot distinguish between the objects in Z_1, t cannot appear in $\mathcal{A}_1$. Similarly, t cannot appear in $\mathcal{A}_2$ for the same reason.

Condition (1) of Definition 1 is easily satisfied by $\mathcal{A}$, since the leaves of $\mathcal{A}$ are labeled by distinct objects from the set $Z_1 \cup Z_2$ due to the fact that $Z_1 \cap Z_2 = \emptyset$.

The nodes in $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the conditions (2), (3), and (4) already, since $\mathcal{A}_1$ and $\mathcal{A}_2$ are decision trees themselves. The tree $\mathcal{A}$ introduces one node only, which is the root. Let r be this root node. Since r is labeled by t, a test in T, condition (2) is satisfied by r. Condition (3) is also satisfied by r, as it has two outgoing edges with label 0 and label 1 (either $x = 0$ and $\bar{x} = 1$, or $x = 1$ and $\bar{x} = 0$).

For condition (4), let us assume that $x = 0$. Let p be a leaf node with label z where $z \in Z_1$. Since $z \in Z_1$, by the premises of the lemma, we have $t(z) = x = 0$. In this case, p is under the 0–successor of r, satisfying the condition (4) for r. For a leaf node p with label z where $z \in Z_2$, by the premises of the lemma, we have $t(z) = \bar{x} = 1$. In this case, p is under the 1–successor of r, satisfying the condition (4) for r, again. For the case where $x = 1$ and $\bar{x} = 0$, the proof is similarly easy. □

Lemma 4. *Let $D[T, Z]$ be a decision table, $z \in Z$ be an object, $T' \subseteq T$ be a minimum verification set for z in $D[T, Z]$, and $t \in T'$ be a test in the minimum verification set T'. Let Z' be the set of objects in Z that give the same response to t as z, i.e. $Z' = \{z' \in Z | t(z') = t(z)\}$. In this case, $T' \setminus \{t\}$ is a minimum verification for z in the decision table $D[T, Z']$.*

Proof. Suppose there exists a verification set T'' for z in $D[T, Z']$ such that $|T''| < |T' \setminus \{t\}|$. Using the tests in T'', z can be distinguished from all the objects in Z'. The test t distinguishes z from all the objects in $Z \setminus Z'$. Hence

the set $T'' \cup \{t\}$ is a verification set for z in $D[T, Z]$. This is a contradiction since $|T'' \cup \{t\}| < |T'|$ and T' is a minimum verification set for z in $D[T, Z]$. □

Lemma 5. *Let $D[T, Z]$ be a decision table, $z \in Z$ be an object, $T' \subseteq T$ be a minimum verification set for z in $D[T, Z]$. There exists a decision tree $\mathcal{A}$ for $D[T, Z]$ such that $\mathcal{A}|_z = T'$.*

Proof. The proof is by induction on the size of the set T'.

When $|T'| = \{t\}$ for a single test t, this means that $t(z) \neq t(z')$, for all $z' \in Z \setminus \{z\}$. Consider the decision table $D[T, Z \setminus \{z\}]$ and let $\mathcal{A}'$ be a decision tree for $D[T, Z \setminus \{z\}]$. Note that t cannot be used in $\mathcal{A}'$, since t cannot distinguish any two objects in $Z \setminus \{z\}$. In this case, using Lemma 3, $\mathcal{A} = \langle z \overset{t(z)}{\leftarrow} t \overset{\overline{t(z)}}{\rightarrow} \mathcal{A}' \rangle$ is a decision tree for $D[T, Z]$. For $\mathcal{A}$, we have $\mathcal{A}|_z = \{t\} = T'$.

For the induction step, let $t \in T'$ be a test in T' and let $Z_1 = \{z' \in Z | t(z') = t(z)\}$, and $Z_2 = Z \setminus Z_1$. Using Lemma 4, $T' \setminus \{t\}$ is a minimum verification set for z in the decision table $D[T, Z_1]$. Since $|T' \setminus \{t\}| < |T'|$, using the induction hypothesis, there exists a decision tree $\mathcal{A}_1$ for $D[T, Z_1]$ such that $\mathcal{A}_1|_z = T' \setminus \{t\}$. Also consider the set of objects Z_2, and let $\mathcal{A}_2$ be a decision tree for $D[T, Z_2]$. Note that $Z_1 \cap Z_2 = \emptyset$, and $\forall z_1, z_1' \in Z_1, \forall z_2, z_2' \in Z_2$, we have $t(z_1) = t(z_1')$, $t(z_2) = t(z_2')$, and $t(z_1) \neq t(z_2)$. Using Lemma 3, $\mathcal{A} = \langle \mathcal{A}_1 \overset{t(z)}{\leftarrow} t \overset{\overline{t(z)}}{\rightarrow} \mathcal{A}_2 \rangle$ is a decision tree for $D[T, Z]$. For $\mathcal{A}$, we have $\mathcal{A}|_z = \mathcal{A}_1|_z \cup \{t\} = (T' \setminus \{t\}) \cup \{t\} = T'$. □

Lemma 5 explains one direction of the connection between MinVS and MinPathDT problems. Namely, for any minimum verification set T', there exists a decision tree $\mathcal{A}$, where only the tests in T' are used in the branch of $\mathcal{A}$ corresponding to the object z. The other direction of the connection is stated by the following lemma.

Lemma 6. *Let $D[T, Z]$ be a decision table, z be an object, and $\mathcal{A}$ be a decision tree. $\mathcal{A}|_z$ is a verification set for z.*

Proof. $\mathcal{A}|_z$ is the set of tests in $\mathcal{A}$ labeling the path from the root to the leaf node p labeled by z. Let p' be another leaf node labeled by another object z'. Consider the node q on the path from the root to p such that p is under x–successor of q (where $x \in \{0, 1\}$) and p' is under $\bar{x}$ successor of q. Since p and p' are two different leaves, we can always find such a node q. Since q is on the path from the root to p, for the label t of q, we know that $t \in \mathcal{A}|_z$. By

condition (4) of Definition 1, we have $t(z) = x \neq \bar{x} = t(z')$, which shows that for t, $t(z) \neq t(z')$. Since for any object $z' \neq z$, we can find such a node q (on the path from root to p) and hence a test $t \in \mathcal{A}|_z$ that distinguishes z from z', $\mathcal{A}|_z$ is a verification set for z. □

Theorem 3. *The decision version of* MinPathDT *problem is* NP-complete.

Proof. Let K be a constant, $D[T, Z]$ be a decision table, z be an object. The decision version of MinPathDT problem asks if there exists a decision tree $\mathcal{A}$ for z in $D[T, Z]$ such that $|d_{\mathcal{A}}(z)| < K$. The size of a decision tree is necessarily polynomially bounded, since each leaf of the tree has a distinct object label. Given a decision tree, one can check the length of the path from the root to the leaf labeled by z to see if the length is smaller than K. Therefore, the decision version of MinPathDT problem is in NP.

For the completeness result, we will use the obvious reduction from the MinVS problem and show that if it is possible to decide MinPathDT problem in polynomial time, it should be possible to decide MinVS problem in polynomial time as well.

Suppose that it is possible to decide if there is a decision tree $\mathcal{A}$ such that $|d_{\mathcal{A}}(z)| < K$ in polynomial time. If such a decision tree $\mathcal{A}$ exists, then using Lemma 6, we can deduce that a verification set $T' = \mathcal{A}|_z$ for z where $|T'| < K$ also exists. If such a decision tree does not exist, then we can also deduce that there is no minimum verification set T' such that $|T'| < K$, since existence of such a set T', would also imply the existence of a decision tree $\mathcal{A}$ where $\mathcal{A}|_z = T'$ using Lemma 5, which is a contradiction.

Therefore, a polynomial time algorithm that can decide MinPathDT problem can also be used to decide MinVS problem. However, by Theorem 1, we know that MinVS problem is NP-complete. □

4.2 Inapproximability of Minimum Path Decision Tree Problem

Due to the construction of our reduction, it can also be shown that the inapproximability result given Section 3.2 for MinVS also applies to MinPathDT. First, we show the relation between the optimal solutions of MinVS and MinPathDT.

Lemma 7. *For a given decision table $D[T, Z]$ and an object z, let OPT_{vs} and OPT_{pdt} be the optimal solutions for* MinVS *and* MinPathDT *problems for $D[T, Z]$ and z, respectively. Then $OPT_{vs} = OPT_{pdt}$.*

Proof. Let T' be a verification set achieving the optimal value OPT_{vs} and $\mathcal{A}$ be a decision tree achieving the optimal value OPT_{pdt}, where $T'' = \mathcal{A}|_z$. Assume that $OPT_{vs} < OPT_{pdt}$, which implies $|T'| = OPT_{vs} < OPT_{pdt} = |T''|$. Lemma 5 claims that there exists another decision tree $\mathcal{A}'$ such that $\mathcal{A}'|_z = T'$. Since $|T'| < |T''|$, this contradicts with the fact that $\mathcal{A}$ is an optimal solution. Reversely, assume that $OPT_{vs} > OPT_{pdt}$, which implies $|T'| = OPT_{vs} > OPT_{pdt} = |T''|$. In this case, using Lemma 6, we know that T'' is also a verification set. However, this cannot happen since T' is a minimum verification set. □

After showing the equivalence of the optimal solutions of these problems, now we can carry the inapproximability result on MinVS to MinPathDT.

Theorem 4. MinPathDT *problem does not admit an $o(\lg n)$ approximation algorithm unless $\mathsf{P} = \mathsf{NP}$.*

Proof. Suppose that $\mathsf{P} \neq \mathsf{NP}$ and there exists a polynomial time algorithm $\mathcal{P}$ such that $\mathcal{P}$ gives on $o(\lg n)$ approximation for MinPathDT. For a given decision table $D[T, Z]$ and an object z, we can then use algorithm $\mathcal{P}$ to find such a decision tree $\mathcal{A}$ in polynomial time. Lemma 5 states that $T' = \mathcal{A}|_z$ is also a verification set for z. According to Lemma 7, the optimal solutions of MinVS and MinPathDT are the same. Hence if $\mathcal{A}$ provides an $o(\lg n)$ approximation to the optimal value of the MinPathDT instance, T' must also provide an $o(\lg n)$ approximation to the MinVS instance. However, due to Theorem 2, we know that $o(\lg n)$ approximation is not possible for MinVS when $\mathsf{P} \neq \mathsf{NP}$. □

5 Conclusion and Future Work

We introduced the problem of verifying an unknown object by using binary tests. Although this is a seemingly easier problem than identifying the unknown object, we showed that the finding a minimum set of tests for such a verification is also NP-complete. In addition, we also proved that minimization version of the problem cannot be approximated within a factor in $o(\lg n)$ unless $\mathsf{P} = \mathsf{NP}$.

We also introduced a new metric that can be used to measure the size of a decision tree. In this new metric, the length of a certain root to leaf path of the decision tree is used as the size of the tree. By showing the equivalence of the problem of finding a minimum verification set for an object and finding a minimum decision tree where the size of the tree is minimized for the root to leaf path of the same object, we showed that decision tree minimization is also NP-complete with respect to this new metric as well. Due to the equivalence of the two problems, the inapproximability result shown for the verification set problem is easily shown to apply for the decision tree minimization problem with this metric as well.

The hardness and the inapproximability results for the verification set problem (and hence for the decision tree minimization problem) are based on a reduction from the set covering problem. There are approximation algorithms for the set covering problem that provide an $O(\lg n)$ approximation. Investigating these algorithms to see if they can be used to provide an $O(\lg n)$ approximation for the problems studied in this work can be an interesting next step.

References

[1] Micah Adler and Brent Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62:1112–1121, 2012.

[2] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, April 2006.

[3] Venkatesan T. Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, and Mukesh Mohania. Decision trees for entity identification: approximation algorithms and hardness results. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '07, pages 53–62. ACM, 2007.

[4] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, July 1998.

[5] M. R. Garey. Optimal binary identification procedures. *SIAM Journal on Applied Mathematics*, 23(2), 1972.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Company, New York, 1979.

[7] L. Hyafil and R. L. Rivest. Constructing Optimal Binary Decision Trees is NP-complete. *Information Processing Letters*, 5:15–17, 1976.

[8] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.

[9] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*. Plenum Press, New York-London, 1972. 85–103.

[10] Eduardo S. Laber and Loana Tito Nogueira. On the hardness of the minimum height decision tree problem. *Discrete Applied Mathematics*, 144(1-2):209–212, November 2004.

[11] D. Lee and M. Yannakakis. Testing finite-state machines: State identification and verification. *IEEE Transactions on Computers*, 43(3):306–320, 1994.

[12] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, September 1994.

[13] Bernard M. E. Moret. Decision trees and diagrams. *ACM Comput. Surv.*, 14(4):593–623, December 1982.

[14] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 475–484, New York, NY, USA, 1997. ACM.