On the hardness of the minimum height decision tree problem

Eduardo S. Laber\textsuperscript{a, 1}, Loana Tito Nogueira\textsuperscript{a}

\textsuperscript{a}Departamento de Informática, PUC-Rio, Informatica, Rua Marques de Sao Vicente 225, R.. 22453900
Rio de Janeiro, Brazil

Received 7 May 2002; received in revised form 31 July 2003; accepted 10 October 2004

Abstract

Given a set of objects $O$ and a set of tests $T$, the abstract decision tree problem (DTP) is to construct a tree with minimum height that completely identifies the objects of $O$, by using the tests of $T$. No algorithm with a good approximation ratio is known to solve this problem. We give a theoretical support for this fact by showing that DTP does not admit an $o(\log n)$-approximation algorithm unless $\text{P} = \text{NP}$.

\textcopyright 2004 Elsevier B.V. All rights reserved.

Keywords: Algorithmical approximation; Computational complexity; Decision trees

1. Introduction

Let $O = \{o_1, o_2, \ldots, o_n\}$ be a set of objects and let $T = \{t_1, \ldots, t_m\}$ be a set of tests. For each test $t_i$, $1 \leq i \leq m$, and each object $o_j$, $1 \leq j \leq n$, we either have $t_i(o_j) = \text{false}$ or $t_i(o_j) = \text{true}$, depending if the object $o_j$ fails in the test $t_i$ or not. We can also think of $t_i$ as a set $t_i = \{o \in O | t_i(o) = \text{true}\}$. The abstract decision tree problem (DTP) is to construct an identification procedure that completely identifies an object of $O$ [5]. This problem can be modeled as an optimization problem on binary trees. Each node of a binary tree corresponds to a test and each leaf to an object. To apply the identification procedure, one first applies the test corresponding to the root of the tree to the unknown object. If it is true, one takes the left branch; otherwise the right one. The procedure is recursively applied to the root of each successive subtree until one reaches a leaf, which identifies the unknown object. The objective function considered here is to minimize the height of the tree, which corresponds to minimizing the number of tests that one needs to perform to identify an object in the worst case.

Fig. 1 shows a decision tree $D$, where the set of objects is $O = \{o_1, o_2, \ldots, o_{14}\}$ and the set of tests is represented by the internal nodes of $D$. For example, $t_1 = \{o_1, o_2, \ldots, o_7\}$, $t_2 = \{o_1, o_2, o_3, o_4\}$, $t_3 = \{o_8, o_9\}$. The caption of this figure should be ignored at this point.

Since this problem is NP-complete [6], one does not expect to find a polynomial time algorithm to solve it. Some heuristical methods are available. In particular, a very natural strategy that always chooses the test that splits the set of objects in the most balanced way is a $\ln n$-approximation algorithm [2]. No algorithm with better performance ratio is known. Here, we give a theoretical support for this fact by showing that this problem does not admit an $O(\log n)$-approximation algorithm unless $\text{NP} = \text{P}$. As an immediate consequence, the greedy algorithm mentioned above is the best, up to constant factors, that one can expect to do in a polynomial time. Though the result presented here is relatively simple, it is relevant. A good indication of this relevance is the large number of papers [1], in different contexts, that cite the NP-completeness of the DTP established by Hyafil and Rivest [6].

\textsuperscript{1}This work was partially supported by FAPERJ through Jovem Cientista Program (E-26/150.362/2002) and by CNPQ, process 300428/99-5.

E-mail addresses: laber@inf.puc-rio.br (E.S. Laber), loana@cos.ufrj.br (L.T. Nogueira).

0166-218X/$ - see front matter \textcopyright 2004 Elsevier B.V. All rights reserved.
Now, we define a reduction.

**Fact 1.** The height of $D(t_i)$ is $\lceil \log |t_i| \rceil$.

Let $OPT_{SC}$ be the size of an optimal solution for $I_{SC}$ and let $OPT_{DT}$ be the height of an optimal tree for the instance $I_{DT}$ obtained through the reduction $\beta$. We have the following lemma.
Lemma 1. \( \text{OPT}_{DT} \leq \text{OPT}_{SC} + \max_{i=1,...,m} \{\lceil \log |t_i| \rceil \} \).

Proof. Let \( \text{OPT}_{SC} = k \) and let \( I^* = \{i_1, i_2, \ldots, i_k\} \) be an optimal solution for \( I_{SC} \). We construct a decision tree \( D \) for \( I_{DT} \) as follows: the rightmost path of \( D \) contains the nodes corresponding to the basis tests \( t_1, \ldots, t_h \) (the order that these tests appear in the rightmost path of \( D \) is not important). For \( j = 1, \ldots, k \), the left subtree rooted on the node \( t_j \) is \( D(t_j) \).

Since the height of \( D(t_j) \) is equal to \( \lceil \log |t_j| \rceil \), which is not larger than \( \max_{i=1,...,m} \{\lceil \log |t_i| \rceil \} \), then the height of \( D \) is at most

\[ k + \max_{i=1,...,m} \{\lceil \log |t_i| \rceil \} = \text{OPT}_{SC} + \max_{i=1,...,m} \{\lceil \log |t_i| \rceil \}. \]

Now, we prove our main theorem, which implies that DTP does not admit an \( o(\log n) \)-approximation algorithm unless \( P = NP \).

Theorem 1. If there is an \( O(\log n) \)-approximation algorithm for DTP, then there is an \( O(\log n) \)-approximation algorithm for the SCP.

Proof. We assume that there is an \( \alpha \)-approximation algorithm, say \( A_{DT} \), to solve the DTP. Moreover, we assume that \( \alpha = o(\log n) \).

We apply the following steps to obtain a solution for \( I_{SC} \).

Step 1: Let \( z \) be a positive integer. Construct a new set covering instance \( I_{SC}', \) containing \( z \) copies of the original set covering instance \( I_{SC} \). More precisely, the new instance has universe set \( U' = \{u'_1, \ldots, u'_n, u''_1, \ldots, u''_n, \ldots, u''_n\} \) and collection set \( C' = \{C_{1z+1}, \ldots, C_{2zm}, C_{2z+1}, \ldots, C_{2iz}, \ldots, C_{(z+1)m}\} \). Furthermore, if \( C_i = \{u_1, \ldots, u_n\} \), then \( C_{pm+i} = \{u_1, \ldots, u_n\} \), for \( p = 1, \ldots, z \).

Step 2: Apply the reduction \( \beta \) to construct an instance \( I_{DT}' \) for DTP starting from \( I_{SC}' \).

Step 3: Use the approximation algorithm \( A_{DT} \) to find a feasible decision tree for \( I_{DT}' \).

Step 4: Construct a solution for \( I_{SC}' \) from the \( I_{DT}' \)'s solution obtained at Step 3 (later, we explain how to do it).

Step 5: Choose the best solution for an individual copy of \( I_{SC} \) from the solution of \( I_{SC}' \) obtained at the previous step. Let \( h \) be the height of the decision tree constructed by \( A_{DT} \) at Step 3. Furthermore, let \( \text{OPT}_{SC}' \) be the value of an optimal solution for \( I_{SC} \), and let \( \text{OPT}_{DT}' \) be the value of an optimal solution for \( I_{DT}' \). We observe that,

\[ h \leq \alpha \text{OPT}_{DT} \leq \alpha (\text{OPT}_{SC} + \lceil \log n \rceil), \]

where the rightmost inequality follows from Lemma 1 and from the fact that every basis test of the instance \( I_{DT}' \) contains at most \( n \) objects.

At Step 4, we construct a solution for \( I_{SC} \) from the solution of \( I_{DT}' \) as follows: Let \( t_1, t_2, \ldots, t_{h'} \) be the tests in the rightmost path in the solution of \( I_{DT}' \). Observe that \( h' \leq h \). For \( i = 1, \ldots, h' \), do: if \( t_i \) is a basis test, then include \( j_i \) in the solution for \( I_{SC} \); otherwise, if \( t_i \) corresponds to an internal node of the decision tree \( D(t_i) \), then include \( p \) in the solution for \( I_{SC} \). Observe that there can be auxiliary tests belonging to trees induced by different basis tests. If this is the case, choose one of them arbitrarily. After that, we must include in the solution of \( I_{SC} \) a set that contains the object corresponding to the right child (if it exists) of \( t_{h'} \).

Let \( I = \{i_1, \ldots, i_{|I|}\} \) be the solution obtained for \( I_{SC} \). Then,

\[ |I| \leq h + 1 \leq \alpha (\text{OPT}_{SC} + \lceil \log n \rceil) + 1. \]

Starting from \( I \), we can easily obtain \( z \) (not necessarily different) feasible solutions for \( I_{SC} \). In fact, for \( p = 1, \ldots, z \), \( I_p = \{j \in |I| \mid p m + 1 \leq j \leq (p + 1) m \} \) is a feasible solution for \( I_{SC} \). Let \( p^* = \arg\min_{p \in \{1, \ldots, z\}} \{|I_p|\} \). At Step 5, the algorithm chooses the solution \( I_{p^*} \). Since \( |I_{p^*}| \leq |I|/z \) and \( \text{OPT}_{SC} = z \text{OPT}_{SC}' \), we have

\[ |I_{p^*}| \leq \alpha \left( \text{OPT}_{SC} + \frac{\lceil \log n \rceil}{z} \right) + 1. \]

By setting \( z = \lceil |I| \rceil \), we obtain that \( |I_{p^*}| \leq \alpha (\text{OPT}_{SC} + 2) \). Since, by assumption, \( \alpha = o(\log n) \), we have

\[ \frac{|I_{p^*}|}{\text{OPT}_{SC}} \leq \frac{\alpha (\text{OPT}_{SC} + 2)}{\text{OPT}_{SC}} = o(\log n). \]

Therefore, we have exhibited an \( o(\log n) \)-approximation algorithm for SCP. \( \square \)
References