Approximating Decision Trees with Multiway Branches

Venkatesan T. Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, and Yogish Sabharwal

IBM India Research Lab, New Delhi, India
{vechakra,pvinayak,sambuddha,ysabharwal}@in.ibm.com

Abstract. We consider the problem of constructing decision trees for entity identification from a given table. The input is a table containing information about a set of entities over a fixed set of attributes. The goal is to construct a decision tree that identifies each entity unambiguously by testing the attribute values such that the average number of tests is minimized. The previously best known approximation ratio for this problem was $O(\log^2 N)$. In this paper, we present a new greedy heuristic that yields an improved approximation ratio of $O(\log N)$.

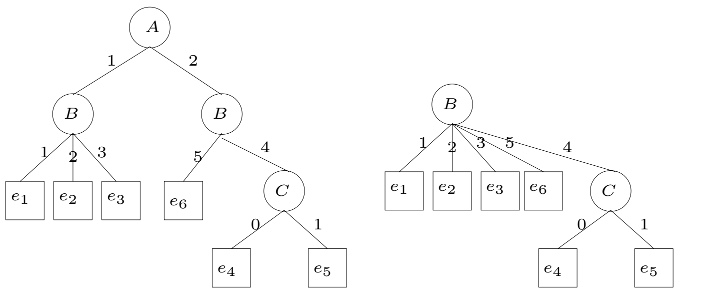
1 Introduction

In many situations (such as machine fault detection, medical diagnosis, species identification), one is required to identify an unknown entity (among a possible set of known entities), by performing a sequence of tests. Typically, each test is restricted to checking the value of an attribute of the unknown entity. Naturally, one wishes to design a testing procedure that minimizes the expected number of tests needed to identify the unknown entity. As an example, consider a typical medical diagnosis application. A hospital maintains a table containing information about diseases. Each row represents a disease and each column is a medical test and the corresponding entry specifies the outcome of the test for a person suffering from the given disease. When the hospital receives a new patient whose disease has not been identified, it would like to determine the shortest sequence of tests which can unambiguously determine the disease of the patient. Motivated by such applications, the problem of constructing *decision trees for entity identification* from given data, has been well studied (See the surveys by Murthy [1] and Moret [2]). In this paper, we present an improved approximation algorithm for the above problem, when the tests are multi-valued.

Decision Trees for Entity Identification - Problem Statement. The input is a table $\mathcal{D}$ having N rows and m columns. Each row is called an *entity* and the columns are the *attributes* of these entities. A solution is a decision tree in which each internal node is labeled by an attribute and its branches are labeled by the values that the attribute can take. The entities are the leaves of the tree. The tree should identify each entity correctly. The goal is to construct a decision tree in which the average distance of a leaf from the root is minimized. Equivalently,

	A	B	C
e_1	1	1	1
e_2	1	2	0
e_3	1	3	1
e_4	2	4	0
e_5	2	4	1
e_6	2	5	0

Input Table



A Decision Tree of cost 14 Optimal Decision Tree of Cost 8

Fig. 1. Example decision trees

we want to minimize the *total external path length* defined as the sum of the distances of all the entities to the root. We call this the $\mathcal{DT}$ problem.

Example: Figure 1 shows an example table and two decision trees. The total external path length of the first tree is 14 and that of the second (optimal) tree is 8. An equivalent way of computing the cost is to sum over the internal nodes v, the number leaves under v. For instance, in the first tree, the cost can be computed as $6 + 3 + 3 + 2 = 14$ and for the second tree, it is $6 + 2 = 8$. □

A special case of the $\mathcal{DT}$ problem is when every attribute can take only two values (say $\{0, 1\}$); this restricted version is called the $2 - \mathcal{DT}$ problem. In general, for some $K \geq 2$, if every attribute is restricted to take at most K distinct values (say $\{0, 1, \ldots, K - 1\}$), we get the $K - \mathcal{DT}$ problem. The general $\mathcal{DT}$ problem has no such restriction on the possible values of attributes.

Hyafil and Rivest [3] showed that the $2 - \mathcal{DT}$ problem is NP-Hard. Garey [4] presented a dynamic programming based algorithm for the $2 - \mathcal{DT}$problem that finds the optimal solution, but the algorithm runs in exponential time in the worst case. Kosaraju et al. [5] studied the $2 - \mathcal{DT}$ problem and showed that a popular greedy heuristic gives an $O(\log N)$ approximation. Adler and Heeringa [6] gave an alternative proof and improved the approximation ratio by a constant factor to $(1 + \ln N)$ (See also [7]).

Garey and Graham [8] showed that the natural greedy algorithm considered in [5,6] has approximation ratio $\Omega(\log N/ \log \log N)$. Chakaravarthy et al. [9] studied the $K - \mathcal{DT}$ problem and gave an $r_K(1 + \ln N)$-approximation algorithm, where r_K is a suitably defined Ramsey number with a known bound that $r_K \leq 2 + 0.64 \log K$; thus, their approximation ratio is $O(\log K \log N)$. On the hardness front, they also showed that the $2 - \mathcal{DT}$ problem and the general $\mathcal{DT}$ problem cannot be approximated within factors of 2 and 4 respectively unless NP=P. The weighted version of the $\mathcal{DT}$ problem has also been studied. Here, there is a probability distribution associated with the set of entities and the goal is to minimize the expected path length. (The unweighted version that we defined corresponds to the case where the distribution is uniform). The $O(\log N)$-approximation algorithm by Kosaraju et al. [5] can also handle the weighted $2-\mathcal{DT}$ problem. For the weighted $K-\mathcal{DT}$ problem, Chakaravarthy et al. [9] gave a $O(\log K \log N)$-approximation algorithm. On the hardness front, it is known that the weighted $2-\mathcal{DT}$ problem cannot be approximated within a factor of $\Omega(\log N)$, unless NP=P [9].

In the general $\mathcal{DT}$ problem, K can be very large (up to N), in which case the approximation ratio of [9] is $O(\log^2 N)$. So, a natural question is whether there exists an approximation algorithm for the general $\mathcal{DT}$ problem matching the approximation ratio of $O(\log N)$ for the $2-\mathcal{DT}$ problem. In this paper, we present an algorithm with an approximation ratio of $O(\log N)$ for the general (unweighted) $\mathcal{DT}$ problem based on a new greedy heuristic.

We now mention the main ingredients of our algorithm and its analysis. Essentially, each internal node of a decision tree narrows down the possibilities for the unknown entity. The greedy heuristic considered in [9], at each internal node, chooses the attribute that maximizes the number of pairs separated (two entities are said to be separated if they do not take the same branch). In contrast, the greedy heuristic that we propose chooses an attribute which minimizes the size of the heaviest branch. In the case of the $2-\mathcal{DT}$ problem, these two heuristics are equivalent. One of the central ideas used in the analysis of [5] is that of "centroidal path". Our main insight lies in drawing a connection between the notion of centroidal path and the objective function of the Minimum Sum Set Cover (MSSC) problem [10]. This insight allows us to utilize some ideas from the analysis of the greedy algorithm for the MSSC problem by Feige et al. [10]. It is interesting to note that the hardness results in [9] are obtained via reductions from the MSSC problem. In this paper, we utilize the MSSC problem in the other direction, namely we use ideas from the analysis of the greedy algorithm for the MSCC problem to obtained improved upper bounds on the approximation ratio.

Our analysis, in its current form, does not extend to the weighted $\mathcal{DT}$ problem. It would be interesting to obtain a $O(\log N)$ approximation algorithm for the weighted $\mathcal{DT}$ problem as that would give matching upper and lower bounds for the problem.

2 Preliminaries

In this section, we define the $\mathcal{DT}$ problem and develop some notations. Let $\mathcal{D}$ be a table having N rows and m columns. Each row is called an *entity* and each column is called an *attribute*. Let $\mathcal{E}$ denote the set of all N entities and $\mathcal{A}$ denote the set of m attributes. For $e \in \mathcal{E}$ and $a \in \mathcal{A}$, denote by $e.a$ the value taken by e on attribute a. We are guaranteed that no two entities are identical. For $a \in A$, let $\mathcal{V}_a$ denote the set of values taken by the attribute a. Let $K = \max_a |\mathcal{V}_a|$. We have $K \leq N$. For instance, if the table is binary, $K = 2$.

A *decision tree* T for the table $\mathcal{D}$ is a rooted tree satisfying the following properties. Each internal node u is labeled by an attribute a and has at most $|\mathcal{V}_a|$ children. Every branch (edge) out of u is labeled by a distinct value from the set $\mathcal{V}_a$. The entities are the leaves of the tree and thus the tree has exactly N leaves. The tree should identify every entity correctly. In other words, for any entity e, the following traversal process should correctly lead to e. The process starts at the root node. Let u be the current node and a be the attribute label of u. Take the branch out of u labeled by $e.a$ and move to the corresponding child of u. The requirement is that this traversal process should reach the entity e.

Observe that the values of the attributes are used only for taking the correct branch during the traversal. So, we can map each value of an attribute to a distinct number from 1 to K and assume that $\mathcal{V}_a$ is a subset of $\{1, 2, \ldots, K\}$. Therefore, we assume that for any $e \in \mathcal{E}$ and $a \in \mathcal{A}$, $e.a \in \{1, 2, \ldots, K\}$.

Let T be a decision tree for $\mathcal{D}$. For an entity $e \in \mathcal{E}$, *path length* of e is defined to be the number of internal nodes in the path from the root to e; it is denoted $\ell_T(e)$ (Note that this is the same as the number of edges on the path). The sum of all path lengths is called *total path length* and is denoted $\mathrm{cost}(T)$, i.e., $\mathrm{cost}(T) = \Sigma_{e \in \mathcal{E}} \ell_T(e)$. We say that e pays a cost of $\ell_T(e)$ in T.

$\mathcal{DT}$ **Problem:** Given a table $\mathcal{D}$, construct a decision tree for $\mathcal{D}$ having the minimum cost.

Remark: In general, we can define a decision tree for any subset of entities of the table ($E \subseteq \mathcal{E}$) and its cost will be defined analogously.

3 Greedy Algorithm

In this section, we describe our greedy algorithm. Let $\mathcal{D}$ be the input table over N entities $\mathcal{E}$ and m attributes $\mathcal{A}$.

Consider a subset of entities $E \subseteq \mathcal{E}$ and an attribute $a \in \mathcal{A}$. The *coverage* of a on E is defined as below. The attribute a partitions the set E into K parts, given by $E_j = \{e \in E \ : \ e.a = j\}$, for $1 \leq j \leq K$. Let j^* be denote the part having the largest size, i.e., $j^* = \mathrm{argmax}_j |E_j|$. We say that *a covers* all the entities in E, except those in E_{j^*} and define

$$\mathrm{Cover}(E, a) = \bigcup_{i \in (\{1,2,\ldots,K\} - j^*)} E_i$$

The entities in E_{j^*} are said to be left *uncovered*. We note that some of the set E_j could be empty.

The greedy algorithm works as follows. For the root of the decision tree, we pick the attribute $\widehat{a}$ having the maximum coverage; equivalently, we minimize the size of the heaviest branch (containing the uncovered entities). The attribute $\widehat{a}$ partitions $\mathcal{E}$ into $E_1, E_2, \ldots E_K$, where $E_i = \{e \ : \ e.\widehat{a} = i\}$. We recursively apply the greedy procedure on each of these subsets and get the required decision tree (See Figure 2). Let $\widetilde{\mathcal{T}}$ denote the decision tree obtained by this procedure. Let $\mathcal{T}_{\mathrm{opt}}$ denote the optimal decision tree for $\mathcal{D}$. We claim that the greedy algorithm has an approximation ratio of $4 \log N$. The theorem is proved in the next section.

Theorem 1. *The greedy algorithm has an approximation ratio of* $4 \log N$*, meaning* $\mathrm{cost}(\widetilde{\mathcal{T}}) \leq (4 \log N)\mathrm{cost}(\mathcal{T}_{\mathrm{opt}})$*.*

```
Procedure Greedy($E$)
Input: $E \subseteq \mathcal{E}$, a set of entities in $\mathcal{D}$;     Output: A decision tree $T$ for the set $E$
Begin
  If $E$ is empty, Return an empty tree.
  If $|E| = 1$,   Return a tree with $x \in E$ as a singleton node.
  Let $\widehat{a}$ be the attribute that covers the maximum number of entities:
    $\widehat{a} = \mathrm{argmax}_{a \in \mathcal{A}} \mathrm{Cover}(E, a)$
  Create the root node $r$ with $\widehat{a}$ as its attribute label.
  For $1 \le j \le K$,
    Let $E_j = \{e \in E | e.\widehat{a} = j\}$
    $T_j = \mathrm{Greedy}(E_j)$
    Let $r_j$ be the root of $T_j$. Add a branch from $r$ to $r_j$ with label $j$ and join $T_j$ to $T$.
  Return $T$ with $r$ as the root.
End
```

Fig. 2. The Greedy Algorithm

4 Analysis of the Greedy Algorithm: Proof of Theorem 1

We extend the notion of cost to all subtrees of the greedy tree $\widetilde{\mathcal{T}}$. Consider an internal node $v \in \widetilde{\mathcal{T}}$ and let $\widetilde{T}_v$ be the subtree rooted at v. Let $L(v)$ denote the entities (or leaves) in $\widetilde{T}_v$; these are the entities that appear below v in $\widetilde{T}$. For instance, if r is the root of $\widetilde{T}$, then $\widetilde{\mathcal{T}} = \widetilde{T}_r$ and $L(r) = \mathcal{E}$. Let $I(\widetilde{T}_v)$ denote the internal nodes of $\widetilde{T}_v$. Define

$$\mathrm{cost}(\widetilde{T}_v) = \sum_{e \in L(v)} (\text{length of path from } v \text{ to } e).$$

Here, length refers to the number of internal nodes in the path from v to e.

Remark: Notice that $\widetilde{T}_v$ is the output of the greedy procedure, if $L(v)$ is the input set of entities. We say that v *handles* the set of entities given by $L(v)$.

Proposition 1. *For any internal node $v \in \mathcal{T}$, $\mathrm{cost}(\widetilde{T}_v) = \Sigma_{x \in I(\widetilde{T}_v)} |L(x)|$.*

Proof. Consider an entity $e \in L(v)$. Let ℓ be the length of the path from v to e; this is the cost contributed by e in $\mathrm{cost}(\mathcal{T}_v)$. Note that there are ℓ internal nodes on this path. We charge a unit cost to each of these internal nodes. Then, the charge accumulated at each internal node x is exactly $|L(x)|$. The proposition follows by summing up these accumulated charges over all the internal nodes of $\widetilde{T}_v$. □

The following proposition is useful in proving Theorem 1.

Proposition 2. *Let $E \subseteq \mathcal{E}$ be a set of entities. Let $E_1, E_2, \ldots, E_d$ be any disjoint subsets of E. Let $T_1^*, T_2^*, \ldots T_d^*$ be the optimal decision trees for $E_1, E_2, \ldots, E_d$, respectively and let T^* be the optimal decision tree for E. Then,*

$$\mathrm{cost}(T_1^*) + \mathrm{cost}(T_2^*) + \cdots + \mathrm{cost}(T_d^*) \le \mathrm{cost}(T^*).$$

Proof. Consider any E_i. We can construct a decision tree T_i for E_i from T^* as follows. Remove from T^* any entity (i.e, leaf) not appearing in E_i; iteratively remove any internal node having no children. Cleary, the resulting tree T_i is a decision tree for E_i. For any entity $e \in E_i$, the cost paid by e in T_i is the same as the cost paid by e in T^*. Since E_i's are disjoint, we have that

$$\mathrm{cost}(T_1) + \mathrm{cost}(T_2) + \cdots + \mathrm{cost}(T_d) \leq \mathrm{cost}(T^*).$$

The lemma now follows directly. □

We shall prove following lemma which generalizes Theorem 1. For a subset of entities $E \subseteq \mathcal{E}$, let T^*_E denote the optimal decision tree for E. Consider an internal node $u \in \widetilde{\mathcal{T}}$. $T^*_{L(u)}$ is the optimal decision tree for $L(u)$. Theorem 1 is obtained by applying the following lemma to the root node r of $\widetilde{\mathcal{T}}$.

Lemma 1. *For any internal node* $u \in \widetilde{\mathcal{T}}$, $\mathrm{cost}(\widetilde{T}_u) \leq (4 \log n)\mathrm{cost}(T^*_{L(u)})$ *where* $n = |L(\widetilde{u})|$.

Proof. The lemma is proved by induction on the number of internal nodes in the subtree under consideration. If $\widetilde{T}_u$ has only one internal node, then each entity pays a cost of 1; this is optimal. Now, let us prove the inductive step.

We state some important definitions first. An internal node x of $\widetilde{\mathcal{T}}$ is said to be *terminal*, if all its children are leaves. For a non-terminal internal node x of $\widetilde{\mathcal{T}}$, its *centroidal child* is defined to be the child of x having the maximum number of leaves (breaking ties arbitrarily). We now extend this to the notion of *centroidal path.* Consider the subtree $\widetilde{T}_v$ rooted at some internal node v of $\widetilde{\mathcal{T}}$. Starting at the root node of $\widetilde{T}_v$ (i.e., the node v), iteratively keep picking the centroidal child till we reach an internal node having only leaves as its children. Such a path is called the centroidal path of $\widetilde{T}_v$ and is denoted by $\mathrm{Cent}(\widetilde{T}_v)$. Cost of the centroidal path is defined as

$$\mathrm{cost}(\mathrm{Cent}(\widetilde{T}_v)) = \sum_{x \in \mathrm{Cent}(\widetilde{T}_v)} |L(x)|.$$

Remark: It is easier to visualize the centroidal path if we make the centroidal child of each node as the right most child. Then, the $\mathrm{Cent}(\widetilde{T}_v)$ will be the right extreme path starting at v.

We now prove the inductive step in Lemma 1. Consider an internal node u. We wish to show that $\mathrm{cost}(\widetilde{T}_u) \leq (4 \log n)\mathrm{cost}(T^*_{L(u)})$, where $n = |L(u)|$

Cost of $\widetilde{T}_u$ can be decomposed into two parts by considering cost incurred along the centroidal path of $\widetilde{T}_u$ and the remaining cost. We say that an internal node v of $\widetilde{T}_u$ is *adjacent* to the centroidal path $\mathrm{Cent}(\widetilde{T}_u)$, if v is not part of the centroidal path, but it is a child of some node in the centroidal path. Let Adj denote the set of all nodes adjacent to $\mathrm{Cent}(\widetilde{T}_u)$. Then, the cost of $\widetilde{T}$ can expressed as below; the expression follows from Proposition 1.

$$\mathrm{cost}(\widetilde{T}_u) = \mathrm{cost}(\mathrm{Cent}(\widetilde{T}_u)) + \sum_{v \in \mathrm{Adj}} \mathrm{cost}(\widetilde{T}_v). \tag{1}$$

Consider any $v \in \text{Adj}$. Since v is a non-centroidal node, we have $|L(v)| \leq n/2$. Moreover, the number of internal nodes in $\widetilde{T}_v$ is less than that of u. Thus, by the inductive hypothesis, we have that

$$\text{cost}(\widetilde{T}_v) \leq 4\log(n/2)T^*_{L(v)}. \tag{2}$$

In Lemma 2 (see Section 5), we show that

$$\text{cost}(\text{Cent}(\widetilde{T}_u)) \leq 4\text{cost}(T^*_{L(u)}) \tag{3}$$

Assuming the above bound, we can complete the proof of Lemma 1. So,

$$\begin{aligned}
\text{cost}(\widetilde{T}_u) &= \text{cost}(\text{Cent}(\widetilde{T}_{L(u)})) + \sum_{v\in\text{Adj}} \text{cost}(\widetilde{T}_v) && \text{(from Eqn. 1)}\\
&\leq 4\text{cost}(T^*_{L(u)}) + \sum_{v\in\text{Adj}} \text{cost}(\widetilde{T}_v) && \text{(from Eqn. 3)}\\
&\leq 4\text{cost}(T^*_{L(u)}) + \sum_{v\in\text{Adj}} 4\log(n/2)\text{cost}(T^*_{L(v)}) && \text{(from Eqn. 2)}\\
&\leq 4\text{cost}(T^*_{L(u)}) + 4\log(n/2)\text{cost}(T^*_{L(u)})\\
& && \text{(from Prop. 2, taking } E = L(u))\\
&\leq 4\log(n)\text{cost}(T^*_{L(u)})
\end{aligned}$$

We have completed the proof of Lemma 1 and hence, Theorem 1, assuming Lemma 2. $\square$

5 Bound on the Centroidal Cost

In this section, we will prove the following lemma.

Lemma 2. *For any internal node* $u \in \widetilde{T}$, $\text{cost}(\text{Cent}(\widetilde{T}_u)) \leq 4\text{cost}(T^*_{L(u)})$.

Proof. Let us start by simplifying the notation. Consider any internal node $u \in \widetilde{T}$. Let $\widetilde{T} = \widetilde{T}_u$ be the greedy subtree rooted at u. and let $E = L(u)$ be the entities handled by $\widetilde{T}$. Notice that $\widetilde{T}$ is nothing but the tree constructed by the greedy algorithm, if E is the set of input entities. Let $T^* = T^*_{L(u)}$ be the optimal tree for E. Our goal is to argue that $\text{cost}(\text{Cent}(\widetilde{T}_u)) \leq 4\text{cost}(T^*)$.

Let r be the length of $\text{Cent}(\widetilde{T})$ and let the nodes on the path be $u_1, u_2, \ldots, u_r$. For $1 \leq i \leq r$, write $E_i = L(u_i)$. Let $a_1, a_2, \ldots a_r$ be the attributes associated with $u_1, u_2, \ldots, u_r$, respectively. Consider a node u_i, for some $1 \leq i < r$. Let $C_i = \text{Cover}(E_i, a_i)$; we say that u_i *covers* the entities in C_i. The node u_r is a boundary case, and is handled separately. The children of u_r are all leaves and these are given by E_r. Recall that $\text{Cover}(E_r, a_r)$ includes all the entities in E_r, except one. For the node u_r, we define $C_r = E_r$. Note that each entity in E is covered by exactly one node among $u_1, u_2, \ldots, u_r$. See Figure 3(a); here, $r = 4$.

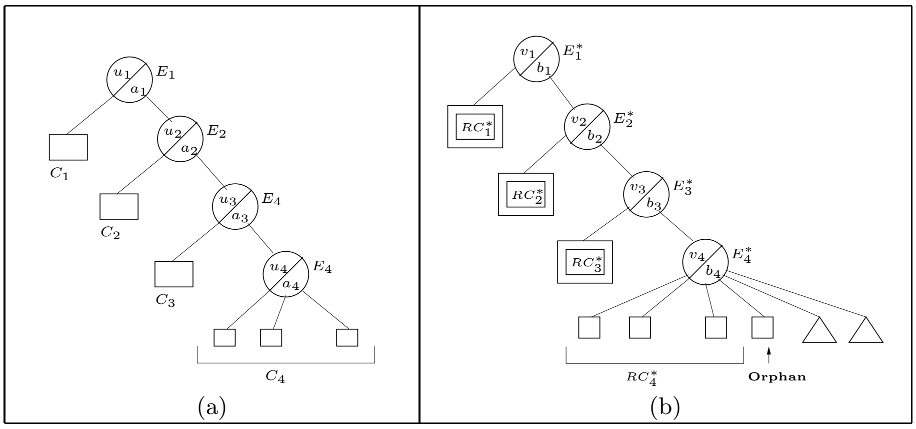


Fig. 3. Centroidal paths of the greedy and the optimal tree

Remark: Each node u_i handles the set of entities given by E_i. The node splits E_i into two parts: the set C_i covered by u_i and the set E_{i+1} handled by u_{i+1}. An alternative way to view $\text{cost}(\text{Cent}(\widetilde{T}))$ is as follows. Each entity in C_i pays a cost of i. Now, $\text{cost}(\text{Cent}(\widetilde{T}))$ is the sum of the cost paid by all the entities, i.e., $\text{cost}(\text{Cent}(\widetilde{T})) = \sum_{i=1}^{r} i|C_i|$. The above function is similar to the objective function in the MSSC problem. This connection allows us to utilize ideas from the analysis of the greedy algorithm for MSSC by Feige et al. [10].

Recall that the cost of the centroidal path $\text{cost}(\text{Cent}(\widetilde{T}))$ is given by $\Sigma_{i=1}^{r}|E_i|$. Consider a node u_i, for some $1 \le i \le r$. We imagine that the node u_i pays a cost of $|E_i|$ towards the centroidal cost. Distribute this cost equally among the entities covered by u_i: for each $e \in C_i$, define its price

$$p_e = \frac{|L(u_i)|}{|C_i|}$$

Notice that

$$\text{cost}(\text{Cent}(\widetilde{T})) = \sum_{e \in E} p_e.$$

Now, let us consider the optimal tree T^*. For each entity $e \in E$, let $\ell^*(e)$ denote length of the path from the root to e in T^*; this is viewed as the cost paid by e in T^*. Recall that $\text{cost}(T^*) = \Sigma_{e \in E} \ell^*(e)$.

We will now bound the price paid by any entity e in the greedy solution $\widetilde{T}$ by the cost paid by some entity e^* in the optimal solution T^* (within a factor of two). For this, we define a mapping $\pi : E \to E$ that maps entities in $\widetilde{T}$ to entities in T^* such that an entity e in $\widetilde{T}$ will charge its price p_e to the entity $\pi(e)$ in T^*. Our mapping π will satisfy the following two properties:

1. For any entity $e^* \in E$, there are at most two entities mapped to it (i.e., $|\{e : \pi(e) = e^*\}| \le 2$).
2. For any entity $e \in E$, $p_e \le 2\ell^*(\pi(e))$.

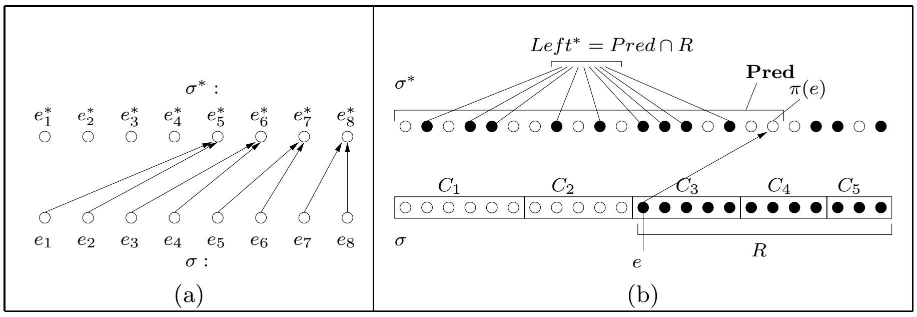


Fig. 4. Illustration of the mapping π and *Left**

We call such a mapping to be *nice*. We shall exhibit a nice mapping π in Section 6. Given such a mapping, we can now see that $\text{cost}(\text{Cent}(\widetilde{T})) \leq 4\text{cost}(T^*)$. This completes the proof of Lemma 2. $\square$

6 A Nice Mapping

In this section, we exhibit a nice mapping π claimed in the proof of Lemma 2.

We arrange the nodes in E in the order in which they are covered by the nodes $u_1, u_2, \ldots, u_r$. Namely, first place all the nodes in C_1 (in some arbitrary order), followed by those in C_2 and so on, finally ending with those in C_r. Let this ordering be denoted $\sigma = e_1, e_2, \ldots e_n$ (where $n = |E|$). In this ordering, the entities come in blocks: first C_1, then C_2 and so on, ending with C_r.

Now, let us consider the optimal tree T^*. Arrange the entities in ascending order of cost $\ell^*(\cdot)$ they pay in T^*. Namely, all the entities that pay a cost of 1 (if any) are placed first (in some arbitrary order), followed by those paying a cost of 2 (if any) and so on. Let this ordering by denoted as $\sigma^* = e_1^*, e_2^*, \ldots, e_n^*$. In σ^*, as we scan from left to right, the cost paid by the entities increases monotonically.

We now define the mapping π as follows. Scan the lists σ and σ^* from right to left, picking two entities from σ and mapping them both to a single entity in σ^*. Formally, if e is an entity appearing in position i from the right in σ, it will be attached to the entity appearing in position $\lceil i/2 \rceil$ from the right in σ^*. See Figure 4(a). Clearly, every entity in σ^* has at most two entities from σ mapped to it. Thus, the mapping π satisfies the Part (1) of the niceness property. Now, we shall show that π also satisfies Part (2) of the niceness property.

6.1 Bounding Greedy Price

In this section, we will show that for any entity e in σ, $p(e) \leq 2\ell^*(\pi(e))$. We will use the following notations.

Consider any subset of entities $S \subseteq \mathcal{E}$. We extend the notion of coverage considering only the entities in S and define the concept of $\text{Cover}_S(\cdot, \cdot)$. Let $X \subseteq \mathcal{E}$ be a subset of entities and a be an attribute. The attribute a partitions the entities X into $X_1, X_2, \ldots, X_K$, where $X_j = \{e \in X \ : \ e.a = j\}$. Let $X_{j'}$ be set among these that has the maximum size, taking only entities in S into account; meaning, let $j' = \text{argmax}_i |X_i \cap S|$. Then, $\text{Cover}_S(X, a)$ is defined to include every set X_i, except $X_{j'}$:

$$\text{Cover}_S(X, a) = \bigcup_{i \in (\{1,2,\ldots,K\} - j')} (X_i \cap S)$$

Consider the optimal tree T^* and $S \subseteq \mathcal{E}$ be some subset of entities. In T^*, we define the notion of *centroidal path with respect to* S. For any internal node $y \in T^*$, let $L(y)$ be the leaves in the subtree rooted at y. By considering only those entities in S, we get $L_S(y)$; i.e., define $L_S(y) = L(y) \cap S$. Consider an internal node $v \in T^*$. Among the children of v, let y be node having the maximum value of $L_S(y)$ (breaking ties arbitrarily); such a y is called the *S-centroidal child of v*. The *S-centroidal path* of T^* is defined next: starting with the root node of T^*, we traverse the tree by picking up the S-centroidal child in each step, until we reach some node y in which the entities in $L_S(y)$ are all children of y.

We are now ready to prove the Part (2) of the niceness property.

Lemma 3. *For any entity e in σ, $\ell^*(\pi(e)) \geq p(e)/2$.*

Proof. Consider some entity e in σ. Let u_h be the node covering e, so that $e \in C_h$. Recall that u_h lies on the centroidal path of $\widetilde{T}$. It handles the entities $E_h = \cup_{i=h}^{r} C_i$. If $h < r$, u_h splits E_h into C_h (which it covers) and $E_{h+1} = \cup_{i=h+1}^{r} C_i$. Let $R = E_h$. Note that $p_e = |R|/|C_h|$.

Let s be the length of R-centroidal path of T^* and let $v_1, v_2, \ldots, v_s$ be nodes appearing on the path. For $1 \leq i \leq s$, let $E_i^* = L(v_i)$ be the entities handled by v_i. Let $b_1, b_2, \ldots, b_s$ be the attributes associated with these nodes. We next define the coverage obtained by each node along the R-centroidal path. For $1 \leq i \leq s$, define $RC_i^* = \text{Cover}_R(E_i^*, b_i)$. Consider $1 \leq t \leq s$. We say that the node v_t *covers* all the entities in RC_t^* and that each entity $\alpha \in RC_t^*$ is said to be covered in the time-step t. The last node v_s presents a boundary case. The entities in $L_R(v_s)$ are all children of v_s; among these all, except one, are included in RC_s^* and covered by v_s; we call the excluded entity as *orphan in* T^*. So, all entities in σ^*, except the orphan, are covered by node v_i. See Figure 3(b); here $s = 4$.

Let *Left** is the entities in R that appear before $\pi(e)$ in σ^*. Formally, let *Pred* be the set of all entities that appear before $\pi(e)$ in σ^* (including $\pi(e)$). Then, *Left** = *Pred* $\cap R$. See Figure 4(b).

Observation 1: Notice that for any entity $e' \in$ *Left**, $\ell^*(e') \leq \ell^*(\pi(e))$ (since, cost of entities in σ^* is monotonically increasing).

We now present an outline of the proof. Intuitively, we will show the following three claims: (i) $|$*Left*$^*| \geq |R|/2$; (ii) an entity covered in time-step t pays a cost of at least t in T^*; (iii) in each time-step t, the optimal tree T^* can cover at most $|C_h|$ entities from R. It would follow that it takes at least $|R|/(2|C_h|)$ time-steps to cover *Left**. Therefore, some entity in *Left** pays a cost of at least

$|R|/(2|C_h|)$. From Observation 1, it follows that $\ell^*(\pi(e)) \geq |R|/(2|C_h|)$. We now formally prove the claims.

Claim 1: $|Left^*| \geq \lceil R/2 \rceil$.

Proof. Since $e \in C_h$ and $R = \cup_{i=h}^{r} C_i$, we have that at most R entities (including e) appear to the right of e in σ. By the definition of the mapping π, we have that in σ^*, at most $\lceil R/2 \rceil$ entities appear to the right of $\pi(e)$ including $\pi(e)$. Thus, in σ^*, at most $\lceil R/2 \rceil - 1$ entities appear to the right of $\pi(e)$ excluding $\pi(e)$. It follows that $|Left^*| \geq \lfloor R/2 \rfloor + 1 \geq \lceil R/2 \rceil$. □

Claim 2: Consider any $1 \leq t \leq s$. Any entity α covered in the time-step t in T^* must satisfy $\ell^*(\alpha) \geq t$.

Proof. To reach to root of T^*, α must climb up along the first t nodes of the R-centroidal path. □

Claim 3: For $1 \leq t \leq s$, at most $|C_h|$ entities from R can be covered in time-step t, i.e, $|RC_t^*| \leq |C_h|$.

Proof. The claim is proved using the fact that the greedy procedure picked attributes that give maximum coverage. Refer to Figure 5 for an illustration. Recall that $L_R(v_t) \subseteq R$ is the entities from R handled by v_t and b_t is the attribute associated with v_t. The attribute b_t partitions $L_R(v_t)$ into $X_1^*, X_2^*, \ldots, X_K^*$, where $X_i^* = \{\alpha \in L_R(v_t) \ : \ \alpha.b_t = i\}$. Let $j^* = \text{argmax}_i |X_i^*|$. Then, note that

$$RC_t^* = \bigcup_{i \in \{1,2,\ldots,K\} - j^*} X_i^*.$$

By contradiction, suppose $|RC_t^*| > |C_h|$. It is easy to see that the union of any $K-1$ distinct sets from $X_1^*, X_2^*, \ldots, X_K^*$ contains at least $|C_h|$ entities. In the greedy tree, imagine what would happen if we picked b_t (instead a_h) as the attribute at the node u_h to split $R = E_h$. Let the partition obtained be $Y_1, Y_2, \ldots, Y_K$, where $Y_i = \{\alpha \in R \ : \ \alpha.b_t = i\}$. So, Cover$(R, b_t)$ is the union of some $K-1$ distinct subsets from $Y_1, Y_2, \ldots, Y_K$. Note that $X_i^* \subseteq Y_i$, for all $1 \leq i \leq K$. Therefore, Cover(R, b_t) is a superset of the union of some $K-1$ subsets from $X_1^*, X_2^*, \ldots, X_K^*$. This implies that $|\text{Cover}(R, b_t)| > |C_h|$. This contradicts the fact that greedy's choice of a_h has the maximum coverage. □

The proof proceeds intuitively as follows. By Claim 3, at most C_h entities from R are covered in each time step. Since $Left^* \subseteq R$, it follows that at most C_h entities in $Left^*$ are covered in each time step. Therefore, it takes at least $|Left^*|/|C_h|$ time-steps to cover all the entities in $Left^*$. Since $|Left^*| \geq \lceil R/2 \rceil$ (by Claim 1), it takes at least $\frac{\lceil R/2 \rceil}{|C_h|}$ steps to cover all the entities in $Left^*$. Thus, by Claim 2, some entity in $Left^*$ pays a cost of at least $\frac{\lceil |R|/2 \rceil}{|C_h|}) \geq \frac{|R|}{2|C_h|}$. From Observation 1, it follows that $\ell^*(\pi(e)) \geq \frac{|R|}{2|C_h|}$.

The only issue in the above argument is that by definition, orphan is not said to be covered by any node. This is a problem if orphan belongs to $Left^*$.

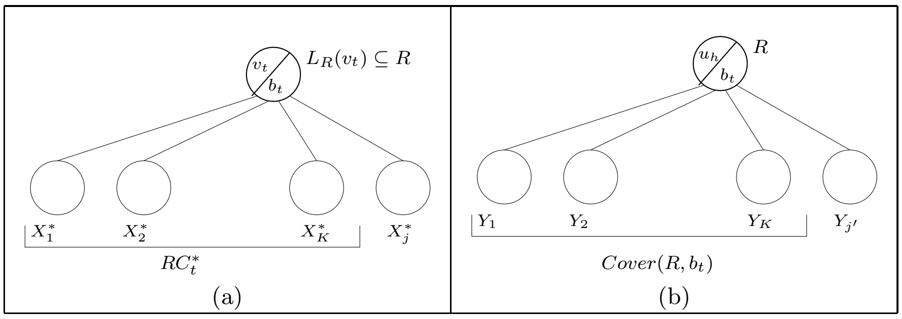


Fig. 5. Illustration for Claim 3 of Lemma 3

However, this needs only a minor argument. Notice that orphan pays a cost of at least s. Recall that any internal node of the R-centroidal path of T^* can cover at most $|C_h|$ entities from R. Therefore at most $s|C_h|$ entities of R can be covered in s time-steps. But, in the s time-steps all the entities in R, except the orphan get covered. Therefore, $s|C_h| \geq R - 1$ and hence, $s \geq (R-1)/|C_h|$. It follows that $s \geq R/(2|C_h|)$, for $R \geq 2$. It is assured that $R \geq 2$, since u_r is an internal node of $\widetilde{T}$ and therefore, has at least two children. Therefore, $s \geq \frac{R}{2|C_h|}$. Thus, orphan pays a cost of at least $\frac{R}{2|C_h|}$. Since orphan is in *Left**, by observation 1, we conclude that $\ell^*(\pi(e)) \geq \frac{|R|}{2|C_h|} = p_e/2$. $\square$

References

1. Murthy, S.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery 2(4), 345–389 (1998)
2. Moret, B.: Decision trees and diagrams. ACM Computing Surveys 14(4), 593–623 (1982)
3. Hyafil, L., Rivest, R.: Constructing optimal binary decision trees is NP-complete. Information Processing Letters 5(1), 15–17 (1976)
4. Garey, M.: Optimal binary identification procedures. SIAM Journal on Applied Mathematics 23(2), 173–186 (1972)
5. Kosaraju, S., Przytycka, M., Borgstrom, R.: On an optimal split tree problem. In: Workshop on Algorithms and Data Structures (1999)
6. Adler, M., Heeringa, B.: Approximating optimal binary decision trees. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX and RANDOM 2008. LNCS, vol. 5171, pp. 1–9. Springer, Heidelberg (2008)
7. Heeringa, B.: Improving Access to Organized Information. Ph.D. thesis, University of Massachusetts, Amherst (2006)
8. Garey, M., Graham, R.: Performance bounds on the splitting algorithm for binary testing. Acta Informatica 3, 347–355 (1974)
9. Chakaravarthy, V., Pandit, V., Roy, S., Awasthi, P., Mohania, M.: Decision trees for entity identification: approximation algorithms and hardness results. In: ACM Symposium on Principles of Database Systems (2007)
10. Feige, U., Lovász, L., Tetali, P.: Approximating min sum set cover. Algorithmica 40(4), 219–234 (2004)