Decision Trees for Entity Identification: Approximation Algorithms and Hardness Results

Venkatesan T. Chakaravarthy Vinayaka Pandit Sambuddha Roy

Pranjal Awasthi Mukesh Mohania

IBM India Research Lab, New Delhi

{vechakra, pvinayak, sambuddha, prawasth, mkmukesh}@in.ibm.com

ABSTRACT

We consider the problem of constructing decision trees for entity identification from a given relational table. The input is a table containing information about a set of entities over a fixed set of attributes and a probability distribution over the set of entities that specifies the likelihood of the occurrence of each entity. 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. This classical problem finds such diverse applications as efficient fault detection, species identification in biology, and efficient diagnosis in the field of medicine. Prior work mainly deals with the special case where the input table is binary and the probability distribution over the set of entities is uniform. We study the general problem involving arbitrary input tables and arbitrary probability distributions over the set of entities. We consider a natural greedy algorithm and prove an approximation guarantee of $O(r_K \cdot \log N)$, where N is the number of entities and K is the maximum number of distinct values of an attribute. The value r_K is a suitably defined Ramsey number, which is at most $\log K$. We show that it is NP-hard to approximate the problem within a factor of $\Omega(\log N)$, even for binary tables (i.e. $K = 2$). Thus, for the case of binary tables, our approximation algorithm is optimal upto constant factors (since $r_2 = 2$). In addition, our analysis indicates a possible way of resolving a Ramsey-theoretic conjecture by Erdös.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms**]: Database Management

General Terms

Algorithms, Theory

Keywords

Decision tree, Ramsey theory, Approximation algorithm

PODS'07, June 11–14, 2007, Beijing, China.

1. INTRODUCTION

Decision trees for the purposes of identification and diagnosis have been studied for a long time now [13]. Consider a typical medical diagnosis application. A hospital maintains a table containing information about diseases. Each row in the table is 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. Some of the medical tests are costly (e.g. MRI scans) and some require few days for the result to be known (e.g. blood cultures). 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. Such a capability would enable it to achieve objectives like saving the expenditure of the patients, quickly determining the disease to start the treatment early etc. Motivated by such applications, we consider the problem of constructing *decision trees for entity identification* from the given data.

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. Additionally, we are also given a probability distribution $\mathcal{P}$ over the set of entities. For each entity e, $\mathcal{P}$ specifies $p(e)$, the likelihood of the occurrence of e. 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 main requirement is that the tree should identify each entity correctly. The cost of the tree is the expected distance of an entity from the root, (i.e., $\Sigma_e p(e)d(e)$, where $d(e)$ is the distance of the entity e from the root). The goal is to construct a decision tree with the minimum cost. We call this the $\mathcal{DT}$ problem.

Figure 1 shows an example table and two decision trees for it. When the probability distribution is uniform, the cost of the first decision tree is $14/6$ and that of the second (and the optimal) decision tree is $8/6$.

For a given table, the maximum number of distinct values that any attribute takes is called its *branching factor*. In the above example, the branching factor of the given table is 5. Interesting special cases of the $\mathcal{DT}$ problem can be obtained in two ways:

- The case in which every input instance has a branching factor of at most K; we call this the K-$\mathcal{DT}$ problem. Of particular interest is the 2-$\mathcal{DT}$ problem, where the tables are binary.

- The case in which the probability distribution over the set of entities is given to be uniform; we call this the $\mathcal{UDT}$ problem.

The special case in which both the above restrictions apply is called the K-$\mathcal{UDT}$ problem.

Previous results. Much of the previous literature deals with the special case of the 2-$\mathcal{UDT}$ problem. Hyafil and Rivest [12] showed that the 2-$\mathcal{UDT}$ problem is NP-hard. Garey [6, 7] presented a dynamic programming based algorithm for the 2-$\mathcal{UDT}$ problem that finds the optimal solution, but the algorithm runs in exponential time in the worst case. Recently, Heeringa and Adler [11] proved the first non-trivial approximation ratio of $(1+\ln N)$ for 2-$\mathcal{UDT}$ (see also [10]). They also showed that it is NP-hard to approximate the 2-$\mathcal{UDT}$ problem within a ratio of $(1+\epsilon)$, for some $\epsilon > 0$. They left open the problem of obtaining an approximation algorithm for 2-$\mathcal{DT}$.

Our results. We study the $\mathcal{DT}$ problem where the attributes can take multiple values and the input probability distribution can be arbitrary. This occurs commonly, for example, in medical diagnosis applications (e.g. blood-group can take multiple values; some diseases are more prevalent than others).

We present an $O(\log N)$-approximation algorithm for the 2-$\mathcal{DT}$ problem. Heeringa and Adler [11] show an $(1+\ln N)$ approximation ratio for the 2-$\mathcal{UDT}$ problem; our analysis builds on their work. We also show that it is NP-hard to approximate the 2-$\mathcal{DT}$ problem within a ratio of $\Omega(\log N)$. Thus, our results for the 2-$\mathcal{DT}$ problem are optimal upto constant factors. For the K-$\mathcal{DT}$ problem, we get an approximation ratio of $O(r_K \log N)$, where r_K is a suitably defined Ramsey number which is at most $(2 + 0.64 \log K)$. For the general $\mathcal{DT}$ problem, the same result applies; here, K refers to the branching factor of the input table.

For the K-$\mathcal{UDT}$ problem we get an approximation ratio of $r_K(1+\ln N)$. We also show that it is NP-hard to approximate the $\mathcal{UDT}$ and the 2-$\mathcal{UDT}$ problems within ratios of $(4-\epsilon)$ and $(2-\epsilon)$ respectively, for any $\epsilon > 0$. The latter result improves the hardness result of Heeringa and Adler [11]. The results are summarized in the following table, where previously known results are cited accordingly.

Problem	Apprx. Ratio	Hardness of Apprx.
2-$\mathcal{UDT}$	$1+\ln N$ [11]	$(2-\epsilon)$ for any $\epsilon > 0$
$\mathcal{UDT}$	$r_K(1+\ln N)$	$(4-\epsilon)$ for any $\epsilon > 0$
2-$\mathcal{DT}$	$O(\log N)$	$\Omega(\log N)$
$\mathcal{DT}$	$O(r_K \log N)$	$\Omega(\log N)$

Ramsey Numbers and Connections to Erdös' Conjecture. As mentioned earlier, our analysis of the approximation algorithms has interesting connections with Ramsey theory and an unresolved conjecture by Erdös. Ramsey theory, treated at length in the book by Graham et. al [9], deals with coloring the edges of complete graphs (or hypergraphs) with a specified number of colors satisfying certain constraints. For our purposes, we need the following specific type of Ramsey numbers.

For $n > 0$, let G_n denote the complete graph on n vertices. A k-coloring of G_n is a coloring of the *edges* of G_n using k colors. For $k > 0$, R_k is defined to be the smallest number n such that any k-coloring of G_n contains a monochromatic triangle [1]. The inverses of the Ramsey numbers are more convenient for our purposes. For $n > 0$, we define r_n to be the smallest number k such that we can color the edges of G_n using only k colors without inducing any monochromatic triangle.

The exact values of the Ramsey numbers for $k > 3$ are not known. However, it is known that for any k, $\frac{3^k+1}{2} \leq R_k \leq 1 + k!e$ (see [24, 15, 23]). Erdös made the conjecture that for some constant α, for all k, $R_k \leq \alpha^k$.

In terms of the inverse Ramsey numbers, the above bounds translate as follows: (i) for any n, $r_n \leq 2{+}0.64 \log n{=}O(\log n)$; (ii) $r_n = \Omega(\log n / \log\log n)$. The Erdös conjecture now reads

$$r_n = \Omega(\log n).$$

Our results provide interesting approaches to address the conjecture. Exhibit a constant $c > 0$ and show that for all $K \geq 2$, it is NP-hard to approximate the K-$\mathcal{DT}$ problem within a factor of $c \log K \log N$. Notice that this would prove the conjecture under the assumption that NP $\neq$ P. Another way of proving the conjecture would be to construct a family of bad instances for our algorithm (which is a simple greedy heuristic). We discuss the details later in the paper.

Applications and Related Work. Decision trees for entity identification (as defined in this paper) have been used for medical diagnosis (as described earlier), species identification in biology, fault detection etc. [13]. Taxonomists release field guides to help identify species based on their characteristics. These guides are often presented in the form of decision trees labeled by species characteristics. Typically, a field biologist identifies the species of a specimen at hand by referring to such guides (hopefully with as few look-ups as possible). Taxonomists refer to such decision trees as "identification keys" and an article on identification keys can be found in [26]. There are several online interactive guides for species identification: Californian lawn weeds [16], Aquatic macroinvertebrates [18], Snakes of Florida [17], South Australian frogs [1]. In fact, computer programs and algorithms for identification and diagnosis applications have been developed for nearly four decades (e.g., [19, 22, 25]).

Murthy [14] and Moret [13] present excellent surveys on the use of decision trees in such diverse fields as machine learning, pattern recognition, taxonomy, switching theory and boolean logic.

2. PRELIMINARIES

In this section, we define the $\mathcal{DT}$ problem and its special cases. We also develop some notation used in the paper.

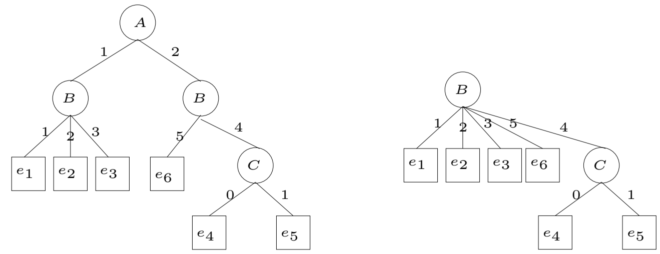
Let $\mathcal{D}$ be a relational table having N tuples and m attributes. We call each tuple an *entity*. Let $\mathcal{E}$ and $\mathcal{A}$ denote the set of entities and attributes, respectively. For $x \in \mathcal{E}$ and $a \in \mathcal{A}$, $x.a$ denotes the value of the entity x on the attribute a. For $a \in \mathcal{A}$, $\mathcal{V}_a$ denotes the set of distinct values taken by a in $\mathcal{D}$. Let $K = \max_{a \in \mathcal{A}}\{|\mathcal{V}_a|\}$. Notice that $K \leq N$. We call K the *branching factor* of $\mathcal{D}$.

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 K children. Every branch (edge) out of u is labeled by a distinct value from the set

[1] A monochromatic triangle is a triplet of vertices such that all the three edges between them have the same color. In Ramsey theory, R_k is denoted $R(3, 3, \ldots, 3)$, where "3" is repeated k times. For example, it is known that $R_1 = 3$, $R_2 = 6$, $R_3 = 17$ [20]

	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 with uniform probability distribution



A Decision Tree of cost 14/6   Optimal Decision Tree of Cost 8/6

Figure 1: Example decision trees

$\mathcal{V}_a$. The entities are the leaves of the tree and thus the tree has exactly N leaves. The main requirement is that the tree should identify every entity correctly. In other words, for any entity x, the following traversal process should correctly lead to x. 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 $x.a$ and move to the corresponding child of u. The requirement is that this traversal process should reach the entity x.

Observe that the values of the attributes are used only for taking the correct branch in the traversal process. 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 $\{1, 2, \ldots, K\}$. In the rest of the paper, we assume that for any $x \in \mathcal{E}$ and $a \in \mathcal{A}$, $x.a \in \{1, 2, \ldots, K\}$.

For a tree T, we use "$u \in T$" to mean that u is an internal node in T. We denote by $\langle x, y\rangle$, an unordered pair of distinct entities.

Let T be a decision tree for $\mathcal{D}$. For an entity $x \in \mathcal{E}$, *path length* of x is defined to be the number of internal nodes in the path from the root to x; it is denoted $\ell_T(x)$. The sum of all path lengths is called *total path length* and is denoted $|T|$, i.e., $|T| = \Sigma_{x\in\mathcal{E}}\ell_T(x)$. Let $w(\cdot)$ be a *weight function* that assigns a real number $w(x) > 0$, for each $x \in \mathcal{E}$. We define the *cost* of T with respect to $w(\cdot)$ as follows:

$$\text{cost}(T, w) = \sum_{x\in\mathcal{E}} w(x)\ell_T(x).$$

We will denote $\text{cost}(T, w)$ as $w(T)$.

As mentioned in the introduction, input to the $\mathcal{DT}$ problem includes a probability distribution $\mathcal{P}$ over $\mathcal{E}$ specifying the likelihood of the occurrence of each entity and the goal is to construct a tree having the minimum expected path length. We view probabilities as weights and assume that the distribution is specified as a weight function $p(\cdot)$ that associates a weight $p(x) > 0$, for each entity x. Notice that when an entity is chosen at random according to the above distribution, the expected path length is given by $p(T) = \text{cost}(T, p)$. We assume that the probabilities $p(x)$ are given as rational numbers. We can easily write these numbers in such way that for any entity x, $p(x) = w(x)/L$, where $w(x) \geq 1$ is an integer and L is an integer giving the common denominator. And so, without loss of generality, we assume the probability distribution will be given as an integer weight function $w(\cdot)$ over a set of entities, i.e, for all $x \in \mathcal{E}$, $w(x) \geq 1$ is an integer. Notice that $p(T) = w(T)/L$ and hence, finding an optimal T under $p(\cdot)$ and $w(\cdot)$ are equivalent.

$\mathcal{DT}$ **Problem.** The input is a relational table $\mathcal{D}$ and a probability distribution $\mathcal{P}$ represented as an integer weight function $w(\cdot)$. The goal is to construct a decision tree T having the minimum cost $w(T)$.

For a positive integer K, the K-$\mathcal{DT}$ problem is a special case of the $\mathcal{DT}$ problem where the input table is required to have a branching factor of at most K. Notice that in the K-$\mathcal{DT}$ problem, the input is a table whose entries are drawn from the set $\{1, 2, \ldots, K\}$.

Of particular interest is the special case called $\mathcal{UDT}$ in which the probability distribution is uniform. In this problem, the weight function is given by $w(x) = 1$, for all $x \in \mathcal{E}$. Note that the cost of a tree T is $w(T) = |T|$. For an integer $K \geq 2$, the special case of the $\mathcal{UDT}$ where the input table is required to have a branching factor of at most K is called the K-$\mathcal{UDT}$ problem.

Remark. Since we are representing probability distributions as integer weight functions, the $\mathcal{UDT}$ and $\mathcal{DT}$ can be thought of as unweighted and weighted versions of the problem respectively.

3. APPROXIMATION ALGORITHMS AND ANALYSIS

In this section, we present an algorithm for the $\mathcal{DT}$ problem and prove an approximation ratio of $O(r_K \log N)$, where K refers to the branching factor of the input table. As mentioned in the introduction, our analysis builds on that of Heeringa and Adler [11] for the 2-$\mathcal{UDT}$ problem. In order to achieve our result, we have to extend their ideas to deal with two issues. Firstly, the attributes can be multi-valued as opposed to binary; secondly, the entities can have arbitrary weights. For ease of exposition, we first show how to address the issue of attributes being multi-valued. Then, we deal with the case of arbitrary weights. Specifically, Section 3.1 presents an algorithm and analysis for the $\mathcal{UDT}$ problem. These ideas are generalized in Section 3.2 to obtain an algorithm for the $\mathcal{DT}$ problem.

3.1 The Unweighted Case: $\mathcal{UDT}$ Problem

This section deals with the $\mathcal{UDT}$ problem. Here, the probability distribution is uniform and so, the weights of all the entities are 1. The goal is to find a tree T with the minimum cost $|T|$.

We present two approximation algorithms for $\mathcal{UDT}$. The first one uses any given α-approximation algorithm for 2-$\mathcal{UDT}$ as a black-box and provides an $\alpha\lceil\log K\rceil$ approximation for the K-$\mathcal{UDT}$ problem. Hence, it has the advantage that any improvement in the approximation ratio for the

2-$\mathcal{UDT}$ problem automatically yields an improvement for the K-$\mathcal{UDT}$ problem. The second one is a greedy heuristic which has an approximation ratio of $r_K(1+\ln N)$. We generalize this procedure to obtain an $O(r_K \log N)$-approximation algorithm for the weighted case of the $\mathcal{DT}$ problem.

3.1.1 The Black-Box Algorithm

We first present a procedure which uses the $(1+\ln N)$-approximation algorithm for 2-$\mathcal{UDT}$ by Heeringa and Adler [11] (referred to as the HA-algorithm) as a black box. The idea is to encode the given $\mathcal{UDT}$ instance as a 2-$\mathcal{UDT}$ instance and then invoke the HA-algorithm on the encoded instance.

Given an $N \times m$ table $\mathcal{D}$ having a branching factor of K, we construct an $N \times m\lceil \log K \rceil$ binary table $\mathcal{D}_2$ as follows. Each attribute in $\mathcal{D}$ is represented by $\lceil \log K \rceil$ attributes in $\mathcal{D}_2$. The former attribute is called the *original* attribute and the latter attributes are called as its *derived* attributes. The values appearing in an original attribute are represented in binary in the corresponding derived attributes. Invoke the HA-algorithm on the binary table $\mathcal{D}_2$ and let $\mathcal{T}_2$ be the decision tree returned by the algorithm. We obtain a decision tree $\mathcal{T}$ for $\mathcal{D}$ from $\mathcal{T}_2$ by replacing the attributes in its internal nodes with their original attributes in $\mathcal{D}$ and labeling appropriately. Notice that $|\mathcal{T}| \le |\mathcal{T}_2|$.

Given a tree T for $\mathcal{D}$, we can construct a tree T_2 for $\mathcal{D}_2$ such that $|T_2| \le \lceil \log K \rceil |T|$. In constructing a decision tree T_2 for the encoded instance $\mathcal{D}_2$, the main task is to take the correct branches of the internal nodes of T using the binary derived attributes. We achieve this by replacing each internal node with a complete binary tree of depth $\lceil \log K \rceil$ using the derived attributes of the original attribute of the internal node. Clearly, $|T_2| \le \lceil \log K \rceil |T|$. This shows that $|\mathcal{T}_2^*| \le \lceil \log K \rceil |\mathcal{T}^*|$ where $\mathcal{T}^*$ and $\mathcal{T}_2^*$ are the optimal decision trees for $\mathcal{D}$ and $\mathcal{D}_2$, respectively. Since $|\mathcal{T}_2| \le (1+\ln N)|\mathcal{T}_2^*|$, the solution $\mathcal{T}$ returned by the black-box algorithm satisfies $|\mathcal{T}| \le \lceil \log K \rceil (1+\ln N)|\mathcal{T}^*|$.

Theorem 3.1. *The black-box algorithm has an approximation ratio of $\lceil \log K \rceil (1+\ln N)$ for the $\mathcal{UDT}$ problem where, K is the branching factor of the input table.*

Notice that we did not make use of any specific properties of the HA-algorithm and so, any other algorithm for the 2-$\mathcal{UDT}$ problem can also be used as the black-box.

3.1.2 The Greedy Algorithm

In this section, we present a greedy algorithm for the $\mathcal{UDT}$ problem. The algorithm is similar in spirit to the HA-algorithm for the 2-$\mathcal{UDT}$ problem. We build on their analysis and develop further combinatorial arguments to obtain our approximation ratio.

Given as input an $N \times m$ table $\mathcal{D}$ having branching factor at most K, the greedy algorithm produces a decision tree $\mathcal{T}$ as described below. Let $\mathcal{E}$ and $\mathcal{A}$ denote the set of entities and attributes of $\mathcal{D}$, respectively. The intuition is that any decision tree should distinguish every pair of distinct entities. So, a natural idea is to make the attribute that distinguishes the maximum number of pairs as the root of $\mathcal{T}$, where an attribute a is said to distinguish a pair $\langle x, y \rangle$, if $x.a \ne y.a$. Choosing such an attribute $\widehat{a}$ can be easily done in time $O(mN^2)$. Picking the attribute $\widehat{a}$ as the label for the root node partitions the set $\mathcal{E}$ into disjoint sets $E_1, E_2, \ldots, E_K$, where $E_i = \{x | x.\widehat{a} = i\}$. We recursively apply the same greedy procedure on each of these sets to obtain K decision trees and make these the subtrees of the root node. The greedy procedure is formally specified in Figure 2. We get the output tree $\mathcal{T}$ by calling $\mathcal{T} = \text{Greedy}(\mathcal{E})$.

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
1. If $|E| = 1$,
   Return a tree with $x \in E$ as a singleton node.
2. Let $\widehat{a}$ be the attribute that distinguishes the maximum number of pairs in E:
   $\widehat{a} = \text{argmax}_{a \in \mathcal{A}} |\{(x,y) | x.a \ne y.a\}|$
3. Create the root node r with $\widehat{a}$ as its attribute label.
4. For $1 \le i \le K$,
   A. Let $E_i = \{x \in E | x.\widehat{a} = i\}$
   B. $T_i = \text{Greedy}(E_i)$
   C. Let r_i be the root of T_i. Add T_i to T by adding a branch from r to r_i with label i.
5. Return T with r as the root.

End

Figure 2: The Greedy Algorithm

Theorem 3.2. *The greedy algorithm has an approximation ratio of $(r_K(1+\ln N))$ for the $\mathcal{UDT}$ problem, where K is the branching factor of the input table.*

We now analyze the greedy algorithm and prove Theorem 3.2. The analysis is divided into two parts. In the first part, we introduce certain combinatorial objects called tabular partitions and analyze the performance of the greedy algorithm using these objects. In the second part, we relate these objects to Ramsey colorings and complete the proof of Theorem 3.2.

3.1.3 Analysis Involving Tabular Partitions

Let $\mathcal{T}$ and $\mathcal{T}^*$ be the greedy and the optimal decision trees, respectively. In this section, we prove a relationship between $|\mathcal{T}|$ and $|\mathcal{T}^*|$ involving tabular partitions, defined below.

Definition 3.3 (Tabular Partitions). *For any positive integer $n \ge 1$, a tabular partition P of n is a sequence $P_1, P_2, \ldots, P_n$ such that P_i is a partition of the set $\{1, 2, \ldots, n\} - \{i\}$. We require that for any distinct $1 \le i, j \le n$, if A is the set in P_i containing j and B is the set in P_j containing i, then $A \cap B = \emptyset$. Let the length of a partition P_i denote the number of sets in it. We define the compactness of P as $\texttt{comp}(P) = \max_i(\text{length of } P_i)$, for $1 \le i \le n$. We define C_n to be the smallest number such that there exists a tabular partition of n having compactness C_n.*

Theorem 3.4. $|\mathcal{T}| \le C_K(1+\ln N)|\mathcal{T}^*|$.

We next focus on proving the above result. In Section 3.1.4, we shall show that $C_K \le r_K$ and obtain Theorem 3.2 by combining the two results. We start with some notations and observations. Let T be any decision tree for $\mathcal{D}$ and u be an internal node of T. We define $E^T(u) \subseteq \mathcal{D}$ to be the set of entities in the subtree of T under u.

Proposition 3.5. *For any decision tree T of $\mathcal{D}$, we have $|T| = \Sigma_{u \in T} |E^T(u)|$.*

Proof. Each entity x contributes a cost equal to its distance from the root. Let us distribute this cost uniformly among the internal nodes on the path from x to the root. Observe that the total cost accumulated at an internal node u is equal to $|E^T(u)|$. Thus, $|T| = \Sigma_{u \in T} |E^T(u)|$. □

Consider a decision tree T and a pair $\langle x, y \rangle$ of entities. We say that a node $u \in T$ *separates* the pair $\langle x, y \rangle$, if the traversal for both x and y passes through u, but x and y take different branches from u. Formally, u is said to separate [2] $\langle x, y \rangle$, if $x, y \in E^T(u)$ and $x.a \neq y.a$, where a is the attribute label of u. For any pair $\langle x, y \rangle$ of entities, there exists a unique separator in T that separates x and y. We define $\mathtt{SEP}(u)$ to be the set of all pairs separated by u. The separators with respect to the greedy tree $\mathcal{T}$ will be important in our analysis. For each pair $\langle x, y \rangle$, we denote by $s_{x,y}$ the separator of $\langle x, y \rangle$ in $\mathcal{T}$ and let $S_{x,y}$ denote $E^{\mathcal{T}}(s_{x,y})$.

From Proposition 3.5, we see that each node $u \in \mathcal{T}$ contributes a cost of $|E^{\mathcal{T}}(u)|$ towards the total cost $|\mathcal{T}|$ and separates the pairs in $\mathtt{SEP}(u)$. We distribute the cost $|E^{\mathcal{T}}(u)|$ equally among the pairs in $\mathtt{SEP}(u)$. For each pair $\langle x, y \rangle \in \mathtt{SEP}(u)$, we define the cost $c_{x,y} = |E^{\mathcal{T}}(u)|/|\mathtt{SEP}(u)|$. Since each pair has a unique separator, the costs $c_{x,y}$ are well-defined.

It is easy to see that $|E^{\mathcal{T}}(u)| = \sum_{\langle x,y \rangle \in \mathtt{SEP}(u)} c_{x,y}$ and by Proposition 3.5, we have $|\mathcal{T}| = \sum_{\langle x,y \rangle} c_{x,y}$, where the summation is taken over all (unordered) pairs of distinct entities. Notice that each pair $\langle x, y \rangle$ also has a unique separator in $\mathcal{T}^*$. So, we rewrite the above summation by partitioning the set of all pairs according to their separators in $\mathcal{T}^*$ and obtain the following equation:

$$|\mathcal{T}| = \sum_{z \in \mathcal{T}^*} \sum_{\langle x,y \rangle \in \mathtt{SEP}(z)} c_{x,y} \tag{1}$$

For each $z \in \mathcal{T}^*$, we define $\alpha(z)$ to be the term corresponding to z in the summation given in Equation 1. Clearly, $\alpha(z) = \sum_{\langle x,y \rangle \in \mathtt{SEP}(z)} c_{x,y}$. The following lemma gives an upperbound on $\alpha(z)$.

Lemma 3.6. *For any $z \in \mathcal{T}^*$, $\alpha(z) \leq C_K(1 + \ln |Z|)|Z|$, where $Z = E^{\mathcal{T}^*}(z)$.*

Assuming the correctness of Lemma 3.6, we first prove Theorem 3.4. The lemma is proved later in the section.
Proof of Theorem 3.4: Replacing the inner summation in Equation 1 by $\alpha(z)$ we have

$$|\mathcal{T}| \leq C_K(1 + \ln N) \sum_{z \in \mathcal{T}^*} |E^{\mathcal{T}^*}(z)| = C_K(1 + \ln N)|\mathcal{T}^*|.$$

The first step is obtained by invoking Lemma 3.6 and the fact that $|Z| \leq N$. Proposition 3.5 gives us the second step.
□

We now proceed to prove Lemma 3.6. Fix any $z \in \mathcal{T}^*$. Let us denote $Z = E^{\mathcal{T}^*}(z)$. Let a_z be the attribute label of z. The node z partitions the set Z into K sets $Z_1, Z_2, \ldots, Z_K$, where $Z_i = \{x \in Z | x.a_z = i\}$. We extend the above notations to sets of values. For any $A \subseteq \{1, 2, \ldots, K\}$, define $Z_A = \cup_{i \in A} Z_i$. We have the following upperbound on $c_{x,y}$ (See Appendix for the proof).

Lemma 3.7. *Let $\langle x, y \rangle \in \mathtt{SEP}(z)$. Consider disjoint sets $A, B \subseteq \{1, 2, \ldots, K\}$ satisfying $y \in Z_A$ and $x \in Z_B$. Then,*

$$c_{x,y} \leq \frac{1}{|S_{x,y} \cap Z_A|} + \frac{1}{|S_{x,y} \cap Z_B|}.$$

[2]We note that the separator of $\langle x, y \rangle$ is nothing but the least common ancestor of x and y.

For each $\langle x, y \rangle$, we shall a choose a suitable pair of disjoint sets A and B and obtain an upperbound on $c_{x,y}$ by invoking Lemma 3.7. We make use of tabular partitions for choosing these sets; the motivation for doing so will become clear in the proof of Lemma 3.10. Let P^* be an optimal tabular partition of K having compactness C_K, given by the sequence $P_1, P_2, \ldots, P_K$. Consider any pair $\langle x, y \rangle \in \mathtt{SEP}(z)$. Let $i = x.a_z$ and $j = y.a_z$ so that $x \in Z_i$ and $y \in Z_j$. Let $\widehat{A}$ be the set in the partition P_i that contains j and $\widehat{B}$ be the set in the partition P_j that contains i. Notice that, by the definition of tabular partitions, the sets $\widehat{A}$ and $\widehat{B}$ are disjoint. We invoke Lemma 3.7 with $\widehat{A}$ and $\widehat{B}$ as the required disjoint sets. (Observe that for any i and j, all the pairs in $Z_i \times Z_j$ will make use of the same disjoint sets while invoking the lemma. Thus the sets chosen depend only on the values $x.a_z$ and $y.a_z$). Therefore,

$$c_{x,y} \leq \frac{1}{|S_{x,y} \cap Z_{\widehat{A}}|} + \frac{1}{|S_{x,y} \cap Z_{\widehat{B}}|}.$$

We split the above cost into two parts and attribute the first term to x and the second term to y. Define

$$c^x_{x,y} = \frac{1}{|S_{x,y} \cap Z_{\widehat{A}}|} \quad \text{and} \quad c^y_{x,y} = \frac{1}{|S_{x,y} \cap Z_{\widehat{B}}|}.$$

It follows that $c_{x,y} \leq c^x_{x,y} + c^y_{x,y}$. For any $x \in Z$, we imagine that x pays a cost $c^x_{x,y}$ to get separated from an entity $y \in Z$. We denote the accumulated cost as $\mathtt{Acc}_z(x)$ and define it as

$$\mathtt{Acc}_z(x) = \sum_{y : \langle x,y \rangle \in \mathtt{SEP}(z)} c^x_{x,y}.$$

Now the lemma given below follows easily.

Lemma 3.8. *For any z, $\alpha(z) \leq \Sigma_{x \in Z} \mathtt{Acc}_z(x)$.*

Our next task is to obtain an upperbound on $\mathtt{Acc}_z(x)$, so that we get a bound on $\alpha(z)$. See Appendix for the proof of the following lemma.

Lemma 3.9. *Let $x \in \mathcal{E}$ be any entity and $Q \subseteq \mathcal{E}$ be any set of entities such that $x \notin Q$. Then,*

$$\sum_{y \in Q} \frac{1}{|S_{x,y} \cap Q|} \leq (1 + \ln |Q|).$$

Lemma 3.10. *For any $x \in Z$, $\mathtt{Acc}_z(x) \leq C_K(1 + \ln |Z|)$*

Proof. Let $r = x.a_z$ and so $x \in Z_r$. Let $\widetilde{Z} = Z - Z_r$ be the rest of the entities in Z. Notice that $\mathtt{Acc}_z(x) = \Sigma_{y \in \widetilde{Z}} c^x_{x,y}$. We perform the above summation by partitioning $\widetilde{Z}$ according to P_r, the r^{th} member of the optimal tabular partition $P^* = P_1, P_2, \ldots, P_K$. Let $P_r = s_1, s_2, \ldots, s_\ell$, where $\ell \leq C_K$. For $1 \leq i \leq \ell$, define $Q_i = \{y \in \widetilde{Z} | y.a_z \in s_i\}$. Thus, $\widetilde{Z} = Q_1 \cup Q_2 \cup \ldots \cup Q_\ell$ and hence,

$$\mathtt{Acc}_z(x) = \sum_{1 \leq i \leq \ell} \sum_{y \in Q_i} c^x_{x,y}. \tag{2}$$

We derive an upperbound for each term in the outer sum using Lemma 3.9. Fix any $1 \leq i \leq \ell$. Notice that for any $y \in Q_i$, we have $c^x_{x,y} = 1/|S_{x,y} \cap Q_i|$, by definition. Moreover, $x \notin Q_i$. Thus, by applying Lemma 3.9 on Q_i, we get

$$\sum_{y \in Q_i} c^x_{x,y} \leq (1 + \ln |Q_i|) \leq (1 + \ln |Z|). \tag{3}$$

We get the lemma by combining Equations 2 and 3, and the fact that $\ell \leq C_K$. $\square$

Proof of Lemma 3.6: The result is proved by combining Lemma 3.8 and Lemma 3.10.

$$\begin{aligned}\alpha(z) &\leq \sum_{x\in Z} \mathtt{Acc}_z(x)\\ &\leq \sum_{x\in Z} C_K(1+\ln|Z|)\\ &= C_K(1+\ln|Z|)|Z|.\end{aligned}$$

$\square$

3.1.4 Tabular Partitions and Ramsey Colorings

In this section, we introduce the notion of directed Ramsey colorings and show that they are equivalent to tabular partitions. Throughout the discussion, for $n > 0$, let G_n and $\widetilde{G}_n$ denote the complete undirected and the complete directed graph on n vertices, respectively.

Definition 3.11. *Let $n > 0$ be an integer. A directed Ramsey coloring of $\widetilde{G}_n$ is a coloring $\widetilde{\tau}$ of the edges such that for any triplet of distinct vertices x, y and z, if $\widetilde{\tau}(x,y) = \widetilde{\tau}(x,z)$ then $\widetilde{\tau}(y,x) \neq \widetilde{\tau}(y,z)$ (and by symmetry, $\widetilde{\tau}(z,x) \neq \widetilde{\tau}(z,y)$).*

We define $\widetilde{R}_k$ to be the smallest number n such that $\widetilde{G}_n$ cannot be directed Ramsey colored using k colors [3]. The inverse of these numbers will be useful. Define $\widetilde{r}_n$ to be the minimum number of colors required to do a directed Ramsey coloring of $\widetilde{G}_n$.

We claim that for any n, there exists a tabular partition P of compactness k if and only if there exists a directed Ramsey coloring $\widetilde{\tau}$ of $\widetilde{G}_n$ that uses only k colors. A proof sketch follows. Let $P = P_1, P_2, \ldots P_n$. Fix $1 \leq x \leq n$. Arrange the sets in the partition P_x in an arbitrary manner, say $P_x = s_{x,1}, s_{x,2}, \ldots, s_{x,\ell}$, where $\ell \leq k$. The n-1 edges outgoing from the vertex x are colored according to the partition P_x. Meaning, for $1 \leq c \leq \ell$, for $y \in s_{x,c}$, we set $\widetilde{\tau}(x,y) = c$. For any y and z, if $\widetilde{\tau}(x,y) = \widetilde{\tau}(x,z)$, then it means that y and z belong to the same set in the partition P_x. By the property of tabular partitions, it should be the case that x and z belong to different sets in the partition P_y, implying that $\widetilde{\tau}(y,x) \neq \widetilde{\tau}(y,z)$. We conclude that $\widetilde{\tau}$ is a directed Ramsey coloring and that $\widetilde{\tau}$ uses only k colors. The converse is proved using a similar argument. The claim implies the following proposition.

Theorem 3.12. *For any n, $C_n = \widetilde{r}_n$.*

Let us call an edge-coloring of G_n a Ramsey coloring, if it does not induce any monochromatic triangles. For any n, a Ramsey coloring τ of G_n readily yields a directed Ramsey coloring $\widetilde{\tau}$ of $\widetilde{G}_n$. For each pair of vertices x and y, we set $\widetilde{\tau}(x,y) = \widetilde{\tau}(y,x) = \tau(x,y)$. It can easily be verified that $\widetilde{\tau}$ is indeed a directed Ramsey coloring of $\widetilde{G}_n$. The number of colors used in $\widetilde{\tau}$ is the same as that of τ. Therefore, we have the following proposition.

Proposition 3.13. *For any n, $\widetilde{r}_n \leq r_n$.*

Proof of Theorem 3.2: The result follows from Theorem 3.4 and 3.12, and Proposition 3.13 $\square$

[3] Such a number exists, as shown in Theorem 5.1

3.2 The Weighted Case: $\mathcal{DT}$ Problem

In this section, we show how to deal with the weighted case, namely the $\mathcal{DT}$ problem. Let D be the input $N \times m$ table over a set of entities $\mathcal{E}$ and a set of attributes $\mathcal{A}$, having a branching factor of K. Let $w(\cdot)$ be the input weight function that assigns an integer weight $w(x) \geq 1$ to each entity $x \in \mathcal{E}$. The problem is to construct an optimal decision tree $\mathcal{T}^*$ having the minimum cost with respect to $w(\cdot)$. We present an algorithm which generalizes the greedy algorithm for the $\mathcal{UDT}$ problem.

Weighted Greedy Algorithm. Refer to the greedy algorithm given in Figure 2. The main step in that algorithm is choosing an attribute that distinguishes the maximum number of pairs. We modify this step so that the weights are taken into account. Namely, we choose the following attribute $\widehat{a}$:

$$\widehat{a} = \text{argmax}_{a\in\mathcal{A}} \sum_{\langle x,y\rangle\in S(a)} w(x)w(y),$$

where $S(a) = \{\langle x,y\rangle | x,y \in \mathcal{E} \text{ and } x.a \neq y.a\}$, is the set of pairs distinguished by the attribute a. We call the above procedure the weighted greedy algorithm.

The following result generalizes Theorem 3.2. Chaudhary and Gupta [2] obtained the same result independently. Let $W = \Sigma_{x\in\mathcal{E}} w(x)$ denote the total weight of the entities. Let $\mathcal{T}$ and $\mathcal{T}^*$ denote the weighted greedy and the optimal trees, under the weight function $w(\cdot)$.

Theorem 3.14. *$w(\mathcal{T}) \leq C_K(1+\ln W)w(\mathcal{T}^*)$, where W is the sum of weights of all the entities.*

We prove the above theorem by adapting the proof of Theorem 3.2. Due to space constraints, we provide an outline of the proof.

Intuitively, we imagine that each entity x is replicated $w(x)$ times and modify the proof of Theorem 3.2 accordingly. We reuse notation from the above proof. Let $\mathtt{SEP}(u)$ be the set of all pairs separated by u. For each pair $\langle x,y\rangle$, we denote by $s_{x,y}$ the separator of $\langle x,y\rangle$ in $\mathcal{T}$ and let $S_{x,y}$ denote $E^{\mathcal{T}}(s_{x,y})$. Additional notation is introduced below.

For a set of entities $X \subseteq \mathcal{E}$, let $w(X)$ denote the total weight of the entities in X, i.e., $w(X) = \Sigma_{x\in X} w(x)$. We also define weights on any set of pairs of entities: for a set of pairs $X \subseteq \mathcal{E}\times\mathcal{E}$, define $w(X) = \Sigma_{\langle x,y\rangle\in X} w(x)w(y)$.

Proposition 3.5 generalizes to the weighted case as follows.

Proposition 3.15. *For a decision tree T of $\mathcal{D}$, $w(T) = \Sigma_{u\in T} w(E^T(u))$.*

For each pair of entities $\langle x,y\rangle$, define a cost $c_{x,y}$ as follows:

$$c_{x,y} = w(x)w(y)\left[\frac{w(S_{x,y})}{w(\mathtt{SEP}(s_{x,y}))}\right].$$

By Proposition 3.15, we get the following equation, which is similar to Equation 1.

$$w(\mathcal{T}) = \sum_{z\in\mathcal{T}^*} \sum_{\langle x,y\rangle\in\mathtt{SEP}(z)} c_{x,y} \qquad (4)$$

For each $z \in \mathcal{T}^*$, the inner summation in Equation 4 is defined as the cost $\alpha(z) = \sum_{\langle x,y\rangle\in\mathtt{SEP}(z)} c_{x,y}$. Our goal is to derive an upperbound on $\alpha(z)$.

Fix any $z \in \mathcal{T}^*$. Let us denote $Z = E^{\mathcal{T}^*}(z)$. Let a_z be the attribute label of z. The node z partitions the set Z into

K sets $Z_1, Z_2, \ldots, Z_K$, where $Z_i = \{x \in Z | x.a_z = i\}$. We extend the above notations to sets of values. For any $A \subseteq \{1, 2, \ldots, K\}$, define $Z_A = \cup_{i \in A} Z_i$. The following lemma generalizes Lemma 3.7 to the weighted case.

Lemma 3.16. *Let $\langle x, y\rangle \in \texttt{SEP}(z)$. Consider disjoint sets $A, B \subseteq \{1, 2, \ldots, K\}$ satisfying $y \in Z_A$ and $x \in Z_B$. Then,*

$$c_{x,y} \leq w(x)w(y)\left[\frac{1}{w(S_{x,y} \cap Z_A)} + \frac{1}{w(S_{x,y} \cap Z_B)}\right].$$

Consider any $\langle x, y\rangle \in \texttt{SEP}(z)$. Let P^* be an optimal tabular partition of K having compactness C_K, given by the sequence $P_1, P_2, \ldots, P_K$. Let $i = x.a_z$ and $j = y.a_z$ so that $x \in Z_i$ and $y \in Z_j$. Let $\widehat{A}$ be the set in the partition P_i that contains j and $\widehat{B}$ be the set in the partition P_j that contains i. Define

$$c^x_{x,y} = \frac{w(x)w(y)}{w(S_{x,y} \cap Z_{\widehat{A}})} \quad \text{and} \quad c^y_{x,y} = \frac{w(x)w(y)}{w(S_{x,y} \cap Z_{\widehat{B}})}.$$

By Lemma 3.16, we have that $c_{x,y} \leq c^x_{x,y} + c^y_{x,y}$. For each entity $x \in E^{\mathcal{T}^*}(z)$, define $\texttt{Acc}_z(x)$ as below:

$$\texttt{Acc}_z(x) = \sum_{y: \langle x,y\rangle \in \texttt{SEP}(z)} c^x_{x,y}.$$

We wish to derive an upperbound on $\texttt{Acc}_z(x)$. The following lemma, which generalizes Lemma 3.9, is useful for this purpose.

Lemma 3.17. *Let $x \in \mathcal{E}$ be any entity and $Q \subseteq \mathcal{E}$ be any set of entities such that $x \notin Q$. Then,*

$$\sum_{y \in Q} \frac{w(y)}{w(S_{x,y} \cap Q)} \leq (1 + \ln w(Q)).$$

The following is obtained by generalizing Lemma 3.10.

Lemma 3.18. *For any $x \in Z$, $\texttt{Acc}_z(x) \leq w(x)C_K(1 + \ln w(Z))$*

Proof of Theorem 3.14: Consider any $z \in \mathcal{T}^*$ and let $Z = E^{\mathcal{T}^*}(z)$. Then, $\alpha(z) \leq \Sigma_{x \in Z}, \texttt{Acc}_z(x)$. Applying Lemma 3.18 and Proposition 3.15, we get that

$$\alpha(z) \leq C_K(1 + \ln w(Z))w(Z). \tag{5}$$

Replacing the inner summation in Equation 4 by $\alpha(z)$ we have

$$\begin{aligned} w(\mathcal{T}) &\leq C_K(1 + \ln W) \sum_{z \in \mathcal{T}^*} w(E^{\mathcal{T}^*}(z)) \\ &= C_K(1 + \ln W)w(\mathcal{T}^*). \end{aligned} \tag{6}$$

The first step is obtained by invoking Equation 5 and the fact that $w(Z) \leq w(\mathcal{E}) = W$. Proposition 3.15 gives us the second step. □

Theorem 3.14 shows that the approximation ratio of the weighted greedy algorithm is logarithmic in N, when the total weight W is polynomially bounded in N. Unfortunately, when the weights are arbitrarily large, the ratio could be worse. We overcome this issue by using the following rounding technique.

Rounded Greedy Algorithm. Let $\mathcal{D}$ be an input table having a branching factor of K and let w_{in} be the input integer weight function. Let $w^{\max}_{\text{in}} = \max_x w_{\text{in}}(x)$ denote the maximum weight. Define a new weight function $w(\cdot)$ as follows: for any entity $x \in \mathcal{E}$, define

$$w(x) = \left\lceil \frac{w_{\text{in}}(x)N^2}{w^{\max}_{\text{in}}} \right\rceil.$$

Run the weighted greedy algorithm with $w(\cdot)$ as the input weight function and obtain a tree $\mathcal{T}$. Return the tree $\mathcal{T}$.

Let $\mathcal{T}^*$ and $\mathcal{T}^*_{\text{in}}$ be the optimal decision trees under the weight functions $w(\cdot)$ and $w_{\text{in}}(\cdot)$, respectively. From Theorem 3.14, we have a good bound for $w(\mathcal{T})$ with respect to $w(\mathcal{T}^*)$. But, of course, we need to compare $w_{\text{in}}(\mathcal{T})$ and $w_{\text{in}}(\mathcal{T}^*_{\text{in}})$. We do this next.

Theorem 3.19. $w_{\text{in}}(\mathcal{T}) \leq 2C_K(1 + 3\ln N)w_{\text{in}}(\mathcal{T}^*_{\text{in}})$.

Proof. Let $x \in \mathcal{E}$ be any entity and consider the path from the root to x in the tree $\mathcal{T}^*_{\text{in}}$. Notice that each internal node along this path separates at least one entity from x. (Otherwise, $\mathcal{T}^*_{\text{in}}$ contains a "dummy" node that does not separate any pairs and hence, can be deleted to obtain a tree of lesser cost). So, the length of the above path is at most N and hence, the following claim is true.

Claim 1: $|\mathcal{T}^*_{\text{in}}| \leq N^2$.

We next compare $w_{\text{in}}(\mathcal{T}^*_{\text{in}})$ and $w(\mathcal{T}^*_{\text{in}})$. We have,

$$\begin{aligned} w(\mathcal{T}^*_{\text{in}}) &= \sum_{x \in \mathcal{E}} w(x)\ell_{\mathcal{T}^*_{\text{in}}}(x) \\ &\leq \sum_{x \in \mathcal{E}} \left(\frac{w_{\text{in}}(x)N^2}{w^{\max}_{\text{in}}} + 1\right)\ell_{\mathcal{T}^*_{\text{in}}}(x) \\ &= \frac{w_{\text{in}}(\mathcal{T}^*_{\text{in}})N^2}{w^{\max}_{\text{in}}} + |\mathcal{T}^*_{\text{in}}| \\ &\leq \frac{w_{\text{in}}(\mathcal{T}^*_{\text{in}})N^2}{w^{\max}_{\text{in}}} + N^2 \\ &\leq \frac{2w_{\text{in}}(\mathcal{T}^*_{\text{in}})N^2}{w^{\max}_{\text{in}}}. \end{aligned} \tag{7}$$

The second step is from the definition of $w(\cdot)$ and the fourth step is obtained from Claim 1. The last inequality is obtained by observing the fact that $w_{\text{in}}(\mathcal{T}^*_{\text{in}}) \geq w^{\max}_{\text{in}}$.

Notice that for any entity $x \in \mathcal{E}$, $1 \leq w(x) \leq N^2$ and so the total weight W under the function $w(\cdot)$ satisfies $W \leq N^3$. So, Theorem 3.14 implies the following claim.

Claim 2: $w(\mathcal{T}) \leq C_K(1 + 3\ln N)w(\mathcal{T}^*)$.

We can now compare $w_{\text{in}}(\mathcal{T})$ and $w_{\text{in}}(\mathcal{T}^*_{\text{in}})$. Note that $\mathcal{T}^*$ is the optimal tree under the function $w(\cdot)$ and hence, $w(\mathcal{T}^*) \leq w(\mathcal{T}^*_{\text{in}})$. We obtain the lemma by combining the observation with Equation 7 and Claim 2. □

By combining Theorem 3.19, Theorem 3.12 and Proposition 3.13, we get the following result.

Theorem 3.20. *The approximation ratio of the rounded greedy algorithm is at most $2r_K(1 + 3\ln N) = O(r_K \log N)$.*

4. HARDNESS OF APPROXIMATION

In this section, we study the hardness of approximating the $\mathcal{DT}$ and the $\mathcal{UDT}$ problems. We show that it is NP-hard to approximate the 2-$\mathcal{DT}$ problem within a ratio of $\Omega(\log N)$. We also improve the previous hardness results for the $\mathcal{UDT}$ problem.

4.1 Hardness of Approximating the 2-$\mathcal{DT}$ Problem

Theorem 4.1. *It is NP-hard to approximate 2-$\mathcal{DT}$ within a factor of $\Omega(\log N)$, where N is the number of entities in the input.*

Proof. We prove the result via a reduction from the set cover problem. It is known that approximating set cover within a factor of $\Omega(\log n)$ is NP-hard [21].

Let $(U, \mathcal{S})$ be the input set cover instance, where $U = \{x_1, x_2, \ldots, x_n\}$ is a universe of items and $\mathcal{S}$ is a collection of sets $\{S_1, S_2, \ldots, S_m\}$ such that $S_i \subseteq U$, for each i. Without loss of generality, we can assume that for any pair of distinct items x_i and x_j, there exists a set $S_k \in \mathcal{S}$ containing exactly one of these two items. (If not, one of these items can be removed from the system.) Construct an instance of the 2-$\mathcal{DT}$ problem having $N = n + 1$ entities and m attributes. The set of entities is $\mathcal{E} = \{x_1, x_2, \ldots, x_n\} \cup \{\widehat{x}\}$, where each entity x_i corresponds to the item x_i and $\widehat{x}$ is a special entity. The set of attributes is $\mathcal{A} = \{S_1, S_2, \cdots, S_m\}$, so that each attribute S_i corresponds to the set S_i. The $N \times m$ table $\mathcal{D}$ is given as follows. For each entity x_i and attribute S_j, set $x_i.S_j = 1$, if $x_i \in S_j$ and otherwise, set $x_i.S_j = 0$. For the special entity $\widehat{x}$, set $\widehat{x}.S_j = 0$, for all attributes S_j. For each entity x_i, set the weight $w(x_i) = 1$. As for the special entity $\widehat{x}$, set its weight as $w(\widehat{x}) = N^3$. This completes the construction.

Let T be a decision tree for $\mathcal{D}$. Let C be the set of attributes found along the path from the root to the entity $\widehat{x}$. Recall that the length of the above path is denoted as $\ell_T(\widehat{x})$. Observe that C is a cover for $(U, \mathcal{S})$. We have $(|C| = \ell_T(\widehat{x})) \leq w(T)/N^3$. On the other hand, given a cover C, we can construct a decision tree T satisfying the following two properties: (i) the set of attributes along the path from the root to $\widehat{x}$ is exactly the set C so that $|\ell_T(\widehat{x})| = |C|$; (ii) for every other entity x_i, $\ell_T(x_i) \leq N$. (The second property is based on the fact that for any table containing N entities, it suffices to test at most N attributes in order to distinguish any entity from the rest). Thus, $w(T) \leq |C|N^3 + N^2$. In particular, $w(T^*) \leq |C^*|N^3 + N^2$, where T^* and C^* are the optimal decision tree and optimal cover, respectively.

Based on the above observations, we can prove the following claim. If there exists an $\alpha(N)$ approximation algorithm for the 2-$\mathcal{DT}$ problem then for any $\epsilon > 0$, we can design an $(1+\epsilon)\alpha(n)$ approximation algorithm for the set cover problem. Therefore, the hardness of set cover problem implies the claimed hardness result for the 2-$\mathcal{DT}$ problem. ☐

4.2 Hardness for the $\mathcal{UDT}$ and the 2-$\mathcal{UDT}$ problems

In this section, we present improved results of hardness of approximation for the $\mathcal{UDT}$ and the 2-$\mathcal{UDT}$ problems. For the 2-$\mathcal{UDT}$ problem, Heeringa and Adler [11] showed a hardness of approximation of $(1+\epsilon)$, for some $\epsilon > 0$. We show that for any $\epsilon > 0$, it is NP-hard to approximate the $\mathcal{UDT}$ and the 2-$\mathcal{UDT}$ problems within a factor of $(4-\epsilon)$ and $(2-\epsilon)$, respectively. Our reductions are from the Minimum Sum Set Cover ($\mathcal{MSSC}$) problem.

The input to the $\mathcal{MSSC}$ problem is a set system: a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ over a universe $U = \{x_1, x_2, \ldots, x_N\}$ of *items*, where each $S_i \subseteq U$. A solution is an ordering π on the sets in $\mathcal{S}$, with an associated cost defined as follows. Let π be $S'_1, S'_2, \ldots, S'_m$. Each item in S'_1 pays a cost of 1, each item in $S'_2 - S'_1$ pays a cost of 2, and so on. Cost of π is the sum of the costs of all items. Formally, define the costs $c^\pi_x = \text{argmin}_i\{x \in S'_i\}$, for $x \in U$, and $\text{cost}(\pi) = \sum_{x \in U} c^\pi_x$. The $\mathcal{MSSC}$ problem is to find an ordering with the minimum cost. For a constant d, the d-$\mathcal{MSSC}$ problem is the special case of $\mathcal{MSSC}$ in which every set in the set system has at most d elements. Feige et. al [5] proved the following hardness results for these problems.

Theorem 4.2. *[5]*

1. *For any $\epsilon > 0$, it is NP-hard to approximate the $\mathcal{MSSC}$ problem within a ratio of $(4 - \epsilon)$.*

2. *For any $\epsilon > 0$, there exists a constant d such that it is NP-hard to approximate d-$\mathcal{MSSC}$ within a ratio of $(2 - \epsilon)$.*

We next prove the hardness result for the $\mathcal{UDT}$ problem by exhibiting a reduction from $\mathcal{MSSC}$. Given an $\mathcal{MSSC}$ instance $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ over a universe $U = \{x_1, x_2, \ldots, x_N\}$, construct an $N \times m$ table $\mathcal{D}$ as follows. Each item x corresponds to an entity and each set S_i corresponds to an attribute a_i. For $1 \leq j \leq m$, $1 \leq i \leq n$, set the entry $x_i.a_j$ as below: if $x_i \in S_j$ then set $x_i.a_j = i$, else set $x_i.a_j = 0$. Observe that any decision tree for $\mathcal{D}$ is *left-deep*: for any internal node u, except the branch labeled 0, every other branch out of u leads to a leaf node.

We claim that given an ordering π of $\mathcal{S}$, we can construct a decision tree $\mathcal{T}$ such that $|\mathcal{T}| = \text{cost}(\pi)$ and vice versa. Let $\pi = S'_1, S'_2, \ldots, S'_m$ and $a'_1, a'_2, \ldots, a'_m$ be the corresponding sequence of attributes. Construct a left-deep tree $\mathcal{T}$, in which the root-node is labeled a'_1 and its 0^{th} child is labeled a'_2 and so on. In general, label the internal node in i^{th} level with a'_i. It can be seen that $\mathcal{T}$ is indeed a decision tree for $\mathcal{D}$ and that $|\mathcal{T}| = \text{cost}(\pi)$. The converse is shown via a similar construction. Given a decision tree $\mathcal{T}$, traverse the tree starting with the root-node and always taking the branches labeled 0. Write down the sequence of sets corresponding to the internal nodes seen in this traversal and let π denote the sequence. Notice that the sets appearing in this sequence cover all elements of U and that $\text{cost}(\pi) = |T|$. (Some sets in $\mathcal{S}$ may not appear in this sequence. To be formally compliant with the definition of solutions, we append the missing sets in an arbitrary order). The claim, in conjunction with Theorem 4.2 (Part 1), implies the following result.

Theorem 4.3. *For any $\epsilon > 0$, it is NP-hard to approximate the $\mathcal{UDT}$ problem within a ratio of $(4 - \epsilon)$.*

The same approach can be used to show the inapproximability of $(2 - \epsilon)$ for the 2-$\mathcal{UDT}$ problem. For this, we consider a reduction from d-$\mathcal{MSSC}$ instances for suitable constants d. Observe that the entries in the table can only be 0 or 1 as opposed to the index of the elements in the previous construction. The required reduction is obtained by using $\lceil \log d \rceil$ auxiliary columns to identify elements of each set. Thus, we get the following result, which improves the hardness result of Heeringa and Adler [11].

Theorem 4.4. *For any $\epsilon > 0$, it is NP-hard to approximate the 2-$\mathcal{UDT}$ problem within a ratio of $(2 - \epsilon)$.*

5. RAMSEY NUMBERS AND ERDOS CONJECTURE

In this section, we take a closer look at our approximation ratio and discuss its connection to a Ramsey-theoretic conjecture by Erdös. We presented an algorithm for the $\mathcal{DT}$ problem having an approximation ratio of $O(r_K \log N)$. Let us now focus on bounds for the inverse Ramsey numbers r_n, for $n \geq 1$.

Recall that for any k, $R_k \geq \frac{3^k+1}{2}$ [15, 23]. From this we get that for any n, $r_n \leq 2 + 0.64 \log n$. Notice that any improvement in the upperbound of r_n would automatically improve our approximation ratio. Better upperbounds are known for r_n (see [15, 4, 3]); but, they improve the above bound only by constant factors. We observe that the upperbound for r_n cannot be improved significantly because of the following result: $R_k \leq 1 + k!e$ [24], which implies $r_n = \Omega(\log n / \log \log n)$.

Observe that our approximation ratio actually involves $\widetilde{r}_n$, rather than r_n. Therefore, one can try to derive a better upperbound on $\widetilde{r}_n$. Unfortunately, we show that $\widetilde{r}_n = \Omega(\log n / \log \log n)$. The claim is implied by the following theorem which can be proved based on an argument similar to the one used to obtain the same bound for R_k.

Theorem 5.1. *For any k, $\widetilde{R}_k \leq 1 + k!e$*

Notice that there is a gap in the upper and lower bounds for R_k. Erdös conjectured that for some constant α, for all k, $R_k \leq \alpha^k$. This is equivalent to $r_n = \Omega(\log n)$.

We discuss the implication of our results in possibly proving the conjecture. The idea is to show that, in terms of worst-case performance factors, the rounded greedy algorithm performs poorly! We observe that a lowerbound of $\Omega(\log K \log N)$ on the approximation ratio for the rounded greedy algorithm would imply the conjecture. More explicitly, we note that the following hypothesis implies the conjecture.

Hypothesis: There exists a constant $\beta > 0$ such that for any K, there exists a K-$\mathcal{DT}$ table $\mathcal{D}$ and a weight function $w(\cdot)$ on which the tree $\mathcal{T}$ produced by the rounded greedy algorithm is such that $w(\mathcal{T}) \geq (\beta \log K \log N) w(\mathcal{T}^*)$, where $\mathcal{T}^*$ is the optimal solution.

A result by Garey and Graham [8] could be a starting point for constructing such instances. They analyzed the worst-case performance of the greedy procedure for the 2-$\mathcal{UDT}$ problem and by constructing counter-examples, obtained a lowerbound of $\Omega(\log N / \log \log N)$ for the approximation ratio of the procedure.

One can also attempt to prove the conjecture under the assumption NP $\neq$ P by showing that it is NP-hard to approximate the K-$\mathcal{DT}$ within a factor of $\Omega(\log K \log N)$. More precisely, exhibit a constant $c > 0$ and show that for all $K \geq 2$, it is NP-hard to approximate the K-$\mathcal{DT}$ problem within a factor of $c \log K \log N$.

6. CONCLUSION AND OPEN PROBLEMS

We studied the problem of constructing good decision trees for entity identification, in the general setup where attributes are multi-valued and the entities are associated with probabilities. We designed an algorithm and proved an approximation ratio involving Ramsey numbers, and also presented hardness results.

There are several interesting open questions. An obvious problem is to design better approximation algorithms and prove better inapproximability bounds for the K-$\mathcal{DT}$ problem.

The directed Ramsey numbers $\widetilde{r}_n$ introduced in this paper pose challenging open problems: Is $\widetilde{r}_n = r_n$, for all n? Is $\widetilde{r}_n = O(\log n / \log \log n)$? Proving the second statement in the affirmative would improve our approximation ratios. If both the statements are shown to be true then the conjecture by Erdös would be disproved! Finally, it would be interesting, if the conjecture can be proved using the approach suggested.

7. ACKNOWLEDGMENTS

We thank Himanshu Gupta and Prasan Roy for useful discussions and comments. We thank the anonymous referees for their insightful suggestions and feedback.

8. REFERENCES

[1] South Australia Environment Protection Authority. Frog identification keys. http://www.epa.sa.gov.au/frogcensus/frog_key.html.

[2] A. Chaudhary and N. Gupta. Personal communication, 2007.

[3] F. Chung and C. Grinstead. A survey of bounds for classical Ramsey numbers. *Journal of Graph Theory*, 7:25–37, 1983.

[4] G. Exoo. A lower bound for Schur numbers and multicolor Ramsey numbers. *Electronic Journal of Combinatorics*, 1(R8), 1994.

[5] U. Feige, L. Lovász, and P. Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

[6] M. Garey. *Optimal binary decision trees for diagnostic identification problems*. Ph.D. thesis, University of Wisconsin, Madison, 1970.

[7] M. Garey. Optimal binary identification procedures. *SIAM Journal on Applied Mathematics*, 23(2):173–186, 1972.

[8] M. Garey and R. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Informatica*, 3:347–355, 1974.

[9] R. Graham, B. Rothschild, and J. Spencer. *Ramsey theory*. John Wiley & Sons, New York, 1990.

[10] B. Heeringa. *Improving Access to Organized Information*. Ph.D. thesis, University of Massachusetts, Amherst, 2006.

[11] B. Heeringa and M. Adler. Approximating optimal decision trees. TR 05-25, University of Massachusetts, Amherst, 2005.

[12] L. Hyafil and R. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[13] B. Moret. Decision trees and diagrams. *ACM Computing Surveys*, 14(4):593–623, 1982.

[14] S. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

[15] J. Nesetril and M. Rosenfeld. I. Schur, C.E. Shannon and Ramsey numbers, a short story. *Discrete Mathematics*, 229(1-3):185–195, 2001.

[16] University of California, Davis. Guide to healthy lawns: Identification key to weeds. http://www.ipm.ucdavis.edu/TOOLS/TURF/PESTS/weedkey.html.
[17] Florida Museum of Natural History, University of Florida. Layman's key to the snakes of florida. http://www.flmnh.ufl.edu/herpetology/FL-GUIDE/snakekey.htm.
[18] The Stream Project, University of Virginia. Aquatic macroinvertebrate identification key. http://wsrv.clas.virginia.edu/∼sos-iwla/Stream-Study/Key/MacroKeyIntro.HTML.
[19] R. Pankhurst. A computer program for generating diagnostic keys. *The Computer Journal*, 13(2):145–151, 1970.
[20] S. Radziszowski. Small Ramsey numbers. *Electronic Journal of Combinatorics*, 1(#7), 1994.
[21] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proc. of the 29th ACM Symposium on Theory of Computing*, pages 475–484, 1997.
[22] A. Reynolds, J. Dicks, I. Roberts, J. Wesselink, B. Iglesia, V. Robert, T. Boekhout, and V. Rayward-Smith. Algorithms for identification key generation and optimization with application to yeast identification. In *EvoWorkshops, LNCS 2611*, pages 107–118, 2003.
[23] I. Schur. Uber die kongruenz $x^m + y^m \equiv z^m \pmod{p}$. *Jber. Deustsch. Math. Verein*, 25:114–117, 1916.
[24] D. West. *Introduction to Graph Theory.* Prentice Hall, 2001.
[25] T. Wijtzes, M. Bruggeman, M. Nout, and M. Zwietering. A computer system for identification of lactic acid bacteria. *International Journal of Food Microbiology*, 38(1):65–70, 1997.
[26] Wikipedia. Identification key — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Dichotomous_key.

APPENDIX

A. PROOFS

A.1 Proof of Lemma 3.7

We are given a pair $\langle x,y\rangle \in \mathtt{SEP}(z)$. Let $s = s_{x,y}$ be the separator of $\langle x,y\rangle$ in $\mathcal{T}$ and the attribute label of s be a_s. The cost $c_{x,y}$ is given by $|S_{x,y}|/|\mathtt{SEP}(s)|$, where $S_{x,y} = E^{\mathcal{T}}(s)$. The greedy algorithm chose the attribute a_s for the node s. Hypothetically, consider choosing the attribute a_z, instead. Let us denote the pairs separated by such a choice as X, i.e., define $X = \{\langle x,y\rangle | x,y \in S_{x,y} \text{ and } x.a_z \neq y.a_z\}$. Notice that the greedy algorithm chose the attribute a_s, instead of a_z, because a_s distinguishes more pairs compared to a_z, meaning, $|\mathtt{SEP}(s)| \geq |X|$. It follows that $c_{x,y} \leq |S_{x,y}|/|X|$. Partition $S_{x,y}$ into $S_1, S_2, \ldots, S_K$, where $S_i = \{x \in S_{x,y} | x.a_z = i\}$.

Then,

$$|X| = \sum_{1\leq i<j\leq K} |S_i|\cdot|S_j|.$$

We bound the summation by considering only the terms $|S_i|\cdot|S_j|$, where $i \in A$ and $j \in B$. Since $A \cap B = \emptyset$ and $|S_{x,y}| = |S_1| + |S_2| + \cdots + |S_K|$, we have that

$$c_{x,y} \leq \frac{1}{\sum_{i\in A}|S_i|} + \frac{1}{\sum_{j\in B}|S_j|}.$$

Observe that for any $1 \leq i \leq K$, $S_{x,y} \cap Z_i \subseteq S_i$ and hence, $|S_{x,y} \cap Z_i| \leq |S_i|$. Therefore,

$$c_{x,y} \leq \frac{1}{\sum_{i\in A}|S_{x,y}\cap Z_i|} + \frac{1}{\sum_{j\in B}|S_{x,y}\cap Z_j|}.$$

Finally, since the sets $Z_{i'}$ and $Z_{j'}$ are disjoint for any distinct $1 \leq i' \leq j' \leq K$, it follows that the first term equals $1/|S_{x,y} \cap Z_A|$ and the second term equals $1/|S_{x,y} \cap Z_B|$. The lemma is proved. □

A.2 Proof of Lemma 3.9

Let $t = |Q|$. We shall prove the following claim:

$$\sum_{y\in Q}\frac{1}{|S_{x,y}\cap Q|} \leq \sum_{i=1}^{t}\frac{1}{i}.$$

The claim implies the lemma, since it is well known that $\sum_{i=1}^{t}(1/i) \leq (1 + \ln t)$, for all t. We prove the claim by applying induction on $|Q|$. For the base case of $|Q| = 1$, let $Q = \{y\}$, where $y \neq x$. Clearly, $y \in S_{x,y}$ and so, $|S_{x,y} \cap Q| = 1$, and the claim follows. Assuming that the claim is true for all sets of size at most $t-1$, we prove it for any set Q of size t. Let y^* be any entity in Q such that for all $y \in Q$, $s_{x,y}$ is a descendent of s_{x,y^*} (a node is considered to be a descendent of itself). If more than one such element exists, pick one arbitrarily. Intuitively, y^* is one among the first batch of entities in Q to get separated from x. The main observation is that $Q \subseteq S_{x,y^*}$ and so, $S_{x,y^*} \cap Q = Q$. Thus, $1/|S_{x,y^*} \cap Q| = 1/|Q| = 1/t$. We apply the induction hypothesis on the set of remaining entities $Q' = Q - y^*$ and infer that

$$\sum_{y\in Q'}\frac{1}{|S_{x,y}\cap Q'|} \leq \sum_{i=1}^{t-1}\frac{1}{i}.$$

Clearly, $Q' \subseteq Q$ and hence, $|S_{x,y} \cap Q'| \leq |S_{x,y} \cap Q|$, So, in the above summation, if we replace the term $|S_{x,y} \cap Q'|$ by $|S_{x,y} \cap Q|$, then the resulting inequality is also true. We conclude that

$$\begin{aligned}\sum_{y\in Q}\frac{1}{|S_{x,y}\cap Q|} &= \frac{1}{|S_{x,y^*}\cap Q|} + \sum_{y\in Q'}\frac{1}{|S_{x,y}\cap Q|}\\ &\leq \frac{1}{t} + \sum_{i=1}^{t-1}\frac{1}{i}\\ &= \sum_{i=1}^{t}\frac{1}{i}.\end{aligned}$$