Approximating Optimal Binary Decision Trees

Micah Adler[1] and Brent Heeringa[2]

[1] Department of Computer Science, University of Massachusetts, Amherst, 140 Governors Drive, Amherst, MA 01003
micah@cs.umass.edu
[2] Department of Computer Science, Williams College, 47 Lab Campus Drive, Williamstown, MA 01267
heeringa@cs.williams.edu

Abstract. We give a $(\ln n + 1)$-approximation for the decision tree (DT) problem. An instance of DT is a set of m binary tests $T = (T_1, \ldots, T_m)$ and a set of n items $X = (X_1, \ldots, X_n)$. The goal is to output a binary tree where each internal node is a test, each leaf is an item and the total external path length of the tree is minimized. Total external path length is the sum of the depths of all the leaves in the tree. DT has a long history in computer science with applications ranging from medical diagnosis to experiment design. It also generalizes the problem of finding optimal average-case search strategies in partially ordered sets which includes several alphabetic tree problems. Our work decreases the previous upper bound on the approximation ratio by a constant factor. We provide a new analysis of the greedy algorithm that uses a simple accounting scheme to spread the cost of a tree among pairs of items split at a particular node. We conclude by showing that our upper bound also holds for the DT problem with weighted tests.

1 Introduction

We consider the problem of approximating optimal binary decision trees. Garey and Johnson [8] define the decision tree (DT) problem as follows: given a set of m binary tests $T = (T_1, \ldots, T_m)$ and a set of n items $X = (X_1, \ldots, X_n)$, output a binary tree where each leaf is labeled with an item from X and each internal node is labeled with a test from T. If an item passes a test it follows the right branch; if it fails a test it follows the left branch. A path from the root to a leaf uniquely identifies the item labeled by that leaf. The depth of a leaf is the length of its path from the root. The total external path length of the tree is the sum of the depths of all the leaves in the tree. The goal of DT is to find a tree which minimizes the total external path length. An equivalent formulation of the problem views each item as an m-bit binary string where bit i is 1 if the item passes test T_i and 0 otherwise. We use instances of this type when discussing DT throughout this paper and denote them using the set of items X. If no two strings in X are identical, every feasible solution to DT has n leaves. In this paper we always assume the input is a set of unique strings since finding duplicate strings is easily computable in polynomial time. Decision trees have many natural applications (see [6,14,17] and references therein) including medical diagnosis (tests are symptoms) and experiment design (tests are experiments which determine some property). In fact, Hyafil and Rivest showed that DT was NP-complete precisely because "of the large

A. Goel et al. (Eds.): APPROX and RANDOM 2008, LNCS 5171, pp. 1–9, 2008.

amount of effort that [had] been put into finding efficient algorithms for constructing optimal binary decision trees" [11].

In this paper, we give a polynomial-time $(\ln n+1)$-approximation for the decision tree problem. This improves the upper bound on the approximation ratio given by Kosaraju et al. [12] by a constant factor. More importantly, our work provides a substantially different analysis of the greedy algorithm for building decision trees. We employ an accounting scheme to spread the total cost of the tree among pairs of items split at internal nodes. The result is an elementary analysis that others may find of independent interest. In fact, our techniques have already been extended to the DT problem with weighted items [4]. We also consider the problem with weights associated with the tests (in contrast to the items) and show that the $(\ln n + 1)$-approximation remains intact.

1.1 Prior and Related Work

DT generalizes the problem of finding optimal search strategies in partially ordered sets when one wishes to minimize the *average* search time (assuming each item is desired with equal probability) as opposed to minimizing the *longest* search time [3]. The latter case corresponds to finding minimal height decision trees. This problem is known to have matching upper and lower bounds ($O(\log n)$ and $\Omega(\log n)$ respectively) on the approximation ratio [2,13,15]. However these results do not generally apply to DT because of the difference in the definition of cost. Additionally, DT generalizes several Huffman coding problems including numerous alphabetic tree problem [12].

The name decision tree also refers to a similar but subtly different problem which we call ConDT (for consistent decision tree) that is extremely hard to approximate. The input to ConDT is a set of n positive / negative labeled binary strings, each of length m, called examples[1]. The output is a binary tree where each internal node tests some bit i of the examples, and maps the example to its left child if i is a 0 and its right child if i is a 1. Each leaf is labeled either TRUE or FALSE. A consistent decision tree maps each positive example to a leaf labeled TRUE and each negative example to a leaf labeled FALSE. The size of a tree is the number of leaves. ConDT seeks the minimum size tree which is consistent with the examples.

Alekhnovich et. al. [1] show it is not possible to approximate size s decision trees by size s^k decision trees for any constant $k \geq 0$ unless NP is contained in DTIME[2^{m^ϵ}] for some $\epsilon < 1$. This improves a result from Hancock et. al. [9] which shows that no $2^{\log^\delta s}$-approximation exists for size s decision trees for any $\delta < 1$ unless NP is quasi-polynomial. These results hold for $s = \Omega(n)$.

Our results demonstrate that DT and ConDT – although closely related – are quite different in terms of approximability: ConDT has no $c \ln n$-approximation for any constant c (unless P = NP) whereas our results yield such an approximation for DT for $c > 1$. Also, we show that the lower bounds on learning decision trees of the ConDT type hold when minimizing total external path length instead of minimum size. Note that tree size is not an insightful measure for DT since all feasible solutions have n leaves. Thus, it is the difference in input and output, and not the difference in measure, that accounts for the difference in approximation complexity.

[1] Many papers take m to be the number of examples and take n to be the number of bits.

Moret [14] views DT and ConDT as unique instances of a general decision tree problem where each item is tagged with k possible labels. With DT there are always $k = n$ labels, but only one item per label. With ConDT, there are only two labels, but multiple items carry the same label. It appears then that labeling restrictions play a crucial role in the complexity of approximating decision trees.

DT shares some similarities with set cover. Since each pair of items is separated exactly once in any valid decision tree, one can view a path from the root to a leaf as a kind of covering of the items. In this case, each leaf defines a set cover problem where it must cover the remaining $n - 1$ items using an appropriate set of bits or tests. In fact, our analysis is inspired by this observation. However, in the decision tree problem, the n set cover problems defined by the leaves are not independent. For example, the bit at the root of an optimal decision tree appears in each of the n set cover solutions, but it is easy to construct instances of DT for which the optimal (independent) solutions to the n set cover instances have no common bits. More specifically, one can construct instances of DT where the n independent set cover problems have solutions of size 1, yielding a decision tree with cost $\Theta(n^2)$ but where the optimal decision tree has cost $O(n \log n)$. Hence, the interplay between the individual set cover problems appears to make the DT problem fundamentally different from set cover. Conversely, set cover instances naturally map to decision tree instances, however, the difference in cost between the two problems means that the optimal set cover is not necessarily the optimal decision tree.

The min-sum set cover (MSSC) problem is also similar to DT. The input to MSSC is the same as set cover (i.e., a universe of items X and a collection C of subsets of X), but the output is a linear ordering of the sets from 1 to $|C|$. If $f(x)$ gives the index of the first set in the ordering that covers x then the cost of the ordering is $\sum_{x \in X} f(x)$. This is similar, but not identical to the cost of the corresponding DT problem because the covered items must still be separated from one another, thus adding additional cost. Greedily selecting the set which covers the most remaining uncovered items yields a 4-approximation to MSSC [5,16]. This approximation is tight unless P=NP. As with set cover, we can think of DT as n instances of MSSC, but again, these instances are not independent so the problems inherent in viewing DT as n set cover problems remain when considering DT as n instances of MSSC.

In the following section we describe and analyze our approximation algorithm for DT. We then extend this analysis to the problem where weights are associated with the tests (but not the items). In Section 3 we show that the lower bounds on learning ConDTs hold for total external path length. Finally, we conclude with a discussion of some open problems including the gap between the upper and lower bounds on the approximation ratio.

2 Approximating DT

Given a set of binary m-bit strings S, choosing some bit i always partitions the items into two sets S^0 and S^1 where S^0 contains those items with bit $i = 0$ and S^1 contains those items with $i = 1$. A greedy strategy for splitting a set S chooses the bit i which minimizes the difference between the size of S^0 and S^1. In other words, it chooses the bit which most evenly partitions the set. Using this strategy, consider the following greedy algorithm for constructing decision trees of the DT type given a set of n items X:

```
GREEDY-DT(X)
1  if X = ∅
2     then return NIL
3     else Let i be the bit which most evenly partitions X into X^0 and X^1
4          Let T be a tree node with left child left[T] and right child right[T]
5          left[T] ← GREEDY-DT(X^0)
6          right[T] ← GREEDY-DT(X^1)
7          return T
```

Fig. 1. A greedy algorithm for constructing decision trees

A straightforward implementation of this algorithm runs in time $O(mn^2)$. While the algorithm does not always give an optimal solution, it does approximate it within a factor of $\ln n + 1$.

Theorem 1. *If X is an instance of DT with n items and optimal cost $\mathcal{C}^*$ then* GREEDY-DT(X) *yields a tree with cost at most* $(\ln n + 1)\mathcal{C}^*$

Proof. We begin with some notation. Let $\mathcal{T}$ be the tree constructed by GREEDY-DT on X with cost $\mathcal{C}$. An unordered pair of items $\{x, y\}$ (hereafter just *pair of items*) is *separated* at an internal node S if x follows one branch and y follows the other. Note that each pair of items is separated exactly once in any valid decision tree. Conversely, each internal node S defines a set $\rho(S)$ of pairs of items separated at S. That is

$$\rho(S) = \{\{x, y\} \mid \{x, y\} \text{ is separated at } S\}$$

For convenience we also use S to denote the set of items in the subtree rooted at S. Let S^+ and S^- be the two children of S such that $|S^+| \geq |S^-|$. Note that $|S| = |S^+| + |S^-|$. The number of sets to which an item belongs equals the length of its path from the root, so the cost of $\mathcal{T}$ may be expressed as the sum of the sizes of each S:

$$\mathcal{C} = \sum_{S \in \mathcal{T}} |S|$$

Our analysis uses an accounting scheme to spread the total cost of the greedy tree among all unordered pairs of items. Since each set S contributes its size to the total cost of the tree, we spread its size uniformly among the $|S^+||S^-|$ pairs of items separated at S. Let c_{xy} be the pair cost assigned to each pair of items $\{x, y\}$ where

$$c_{xy} = \frac{1}{|S^+_{xy}|} + \frac{1}{|S^-_{xy}|}.$$

and S_{xy} separates x from y. Note that the greedy choice minimizes c_{xy}. We can now talk about the cost of a tree node S by the costs associated with the pairs of items separated at S. Summing the costs of these pairs is, by definition, exactly the size of S:

$$\sum_{\{x,y\} \in \rho(S)} c_{xy} = |S^+||S^-|\Big(\frac{1}{|S^+|} + \frac{1}{|S^-|}\Big) = |S|$$

Because two items are separated exactly once, $\mathcal{C}$ is exactly the sum of the all pair costs

$$\mathcal{C} = \sum_{\{x,y\}} c_{xy}.$$

Now consider the optimal tree $\mathcal{T}^*$ for X. If Z is an internal node of $\mathcal{T}^*$ then we also use Z to denote the set of items that are leaves of the subtree rooted at Z. Following our notational conventions, we let Z^+ and Z^- be the children of Z such that $|Z^+| \geq |Z^-|$ and $|Z| = |Z^+| + |Z^-|$. The cost of the optimal tree, $\mathcal{C}^*$, is

$$\mathcal{C}^* = \sum_{Z \in \mathcal{T}^*} |Z| \tag{1}$$

Since, every feasible tree separates each pair of items exactly once, we can rearrange the greedy pair costs according to the structure of the optimal tree:

$$\mathcal{C} = \sum_{Z \in \mathcal{T}^*} \sum_{\{x,y\} \in \rho(Z)} c_{xy} \tag{2}$$

If Z is a node in the optimal tree, then it defines $|Z^+||Z^-|$ pairs of items. Our goal is to show that the sum of the c_{xy} associated with the $|Z^+||Z^-|$ pairs of items split at Z (but which are defined with respect to the greedy tree) total at most a factor of $H(|Z|)$ more than $|Z|$ where $H(d) = \sum_{i=1}^{d} 1/i$ is the d^{th} harmonic number. This is made precise in the following lemma:

Lemma 1. *For each internal node Z in the optimal tree:*

$$\sum_{\{x,y\} \in \rho(Z)} c_{xy} \leq |Z| H(|Z|)$$

where each c_{xy} is defined with respect to the greedy tree $\mathcal{T}$.

Proof. Consider any node Z in the optimal tree. For any unordered pair of items $\{x, y\}$ split at Z, imagine using the bit associated with the split at Z on the set S_{xy} separating x from y in the greedy tree. Call the resulting two sets $S_{xy}^{Z^+}$ and $S_{xy}^{Z^-}$ respectively. Since the greedy split at S_{xy} minimizes c_{xy}, we know

$$c_{xy} = \frac{1}{|S_{xy}^+|} + \frac{1}{|S_{xy}^-|} \leq \frac{1}{|S_{xy}^{Z^+}|} + \frac{1}{|S_{xy}^{Z^-}|} \leq \frac{1}{|S_{xy} \cap Z^+|} + \frac{1}{|S_{xy} \cap Z^-|}.$$

Hence

$$\sum_{\{x,y\} \in \rho(Z)} c_{xy} \leq \sum_{\{x,y\} \in \rho(Z)} \frac{1}{|S_{xy} \cap Z^+|} + \frac{1}{|S_{xy} \cap Z^-|}. \tag{3}$$

One interpretation of the sum in (3) views each item x in Z^+ as contributing

$$\sum_{y \in Z^-} \frac{1}{|S_{xy} \cap Z^-|}$$

to the sum and each node y in Z^- as contributing

$$\sum_{x \in Z^+} \frac{1}{|S_{xy} \cap Z^+|}$$

to the sum. For clarity, we can view Z as a complete bipartite graph where the items in Z^+ are one set of nodes and the items in Z^- is the other set. Letting $b_{xy} = 1/(|(S_{xy} \cap Z^-|)$ and $b_{yx} = 1/(|S_{xy} \cap Z^+|)$ we can think of every edge (x, y) where $x \in Z^+$ and $y \in Z^-$ as having two costs: one associated with x (b_{xy}) and the other associated with y (b_{yx}). Thus, the cost of Z is at most the sum of all the b_{xy} and b_{yx} costs. We can bound the total cost by first bounding all the costs associated with a particular node. In particular, we claim:

Claim. For any $x \in Z^+$ we have

$$\sum_{y \in Z^-} b_{xy} = \sum_{y \in Z^-} \frac{1}{|S_{xy} \cap Z^-|} \leq H(|Z^-|)$$

Proof. If Z^- has d items then let $(y_1, \ldots, y_d)$ be an ordering of Z^- in reverse order from when the items are split from x in the greedy tree (with ties broken arbitrarily). This means item y_1 is the last item split from x, y_d is the first item split from x, and in general y_{d-t+1} is the t^{th} item split from x. When y_d is split from x there must be at least $|Z^-|$ items in S_{xy_d} — by our ordering the remaining items in Z^- must still be present — so $Z^- \subseteq S_{xy_d}$. Hence b_{xy_d}, the cost assigned to x on the edge (x, y_d), is at most $1/|(Z^-)|$ and in general, when y_t is separated from x there are at least t items remaining from Z^-, so the cost b_{xy_t} assigned to the edge (x, y_t) is at most $1/t$. This means, for any $x \in Z^+$

$$\sum_{y \in Z^-} b_{xy} \leq H(|Z^-|)$$

which proves the claim. □

We can use the same argument to prove the analogous claim for all the items in Z^-. With these inequalities in hand we have

$$\begin{aligned}\sum_{\{x,y\} \in \rho(Z)} \frac{1}{|S_{xy} \cap Z^+|} + \frac{1}{|S_{xy} \cap Z^-|} &\leq |Z^+|H(|Z^-|) + |Z^-|H(|Z^+|) \\ &< |Z^+|H(|Z|) + |Z^-|H(|Z|) \\ &= |Z|H(|Z|) \qquad \text{(since } |Z^+| + |Z^-| = |Z|)\end{aligned}$$

□

Substituting this result into the initial inequality completes the proof of the theorem.

$$\sum_{Z \in \mathcal{T}^*} \sum_{\{x,y\} \in \rho(Z)} c_{xy} \leq \sum_{Z \in \mathcal{T}^*} |Z|H(|Z|) \leq \sum_{Z \in \mathcal{T}^*} |Z|H(n) = H(n)\mathcal{C}^* \leq (\ln n + 1)\mathcal{C}^*$$

□

2.1 Tests with Weights

In many applications, different tests may have different execution costs. For example, in experiment design, a single test might be a good separator of the items, but it may also be expensive. Running multiple, inexpensive tests may serve the same overall purpose, but at less cost. To model scenarios like these we associate a weight $w(k)$ with each bit k and without confusion take $w(S)$ to be the weight of the bit used at node S. We call this problem DT with weighted tests (in contrast to the DT problem with weighted items). In the original problem formulation, we can think of each test as having unit weight, so the cost of identifying an item is just the length of the path from the root to the item. When the tests have non-uniform weights, the cost of identifying an item is the sum of the weights of the tests along that path. We call this the path cost. The cost of the tree is the sum of the path costs of each item. When all the tests have equal weight, we choose the bit which most evenly splits the set of items into two groups. In other words, we minimize the pair cost c_{xy}. With equal weights, the cost of an internal node is just its size $|S|$. With unequal weights, the cost of an internal node is the weighted size $w(S)|S|$, so assuming S separates x from y the pair cost becomes

$$c_{xy} = \frac{w(S)}{|S^+|} + \frac{w(S)}{|S^-|} \tag{4}$$

and our new greedy algorithm recursively selects the bit which minimizes this quantity. This procedure yields a result equivalent to Theorem 1 for DT with weighted tests. A straightforward implementation on this algorithm still runs in time $O(mn^2)$.

Theorem 2. *The greedy algorithm which recursively selects the bit that minimizes Equation 4 yields a* $(\ln n + 1)$*-approximation to DT with weighted tests.*

Proof. Following the structure of the proof for Theorem 1 leads to the desired result. The key observation is that choosing the bit that minimizes Equation 4 yields the inequality

$$c_{xy} \leq w(Z)\Big(\frac{1}{|S_{xy} \cap Z^+|} + \frac{1}{|S_{xy} \cap Z^-|}\Big). \tag{5}$$

Since the weight term $w(Z)$ may be factored out of the summation

$$w(Z) \sum_{\{x,y\} \in \rho(Z)} \frac{1}{|S_{xy} \cap Z^+|} + \frac{1}{|S_{xy} \cap Z^-|}$$

we can apply the previous claim and the theorem follows:

$$\sum_{Z \in \mathcal{T}^*} \sum_{\{x,y\} \in \rho(Z)} c_{xy} \leq \sum_{Z \in \mathcal{T}^*} w(Z)|Z|H(n) \leq (\ln n + 1)\mathcal{C}^*$$

Here $C^* = \sum_{Z \in \mathcal{T}^*} w(Z)|Z|$ is the cost of the optimal tree. □

Another natural extension to DT considers the problem with weighted items. Here, one weights each path length by the weight of the item which defines the path. Recently, Chakaravarthy et al. [4] extended our analysis to the DT problem with weighted items.

3 Hardness of Approximation for ConDT under Total External Path Length

Alekhnovich et. al. [1] showed it is not possible to approximate size s decision trees by size s^k decision trees for any constant $k \geq 0$ unless NP is contained in DTIME[2^{m^ϵ}] for some $\epsilon < 1$. Decision tree here refers to trees of the ConDT type and the measure is tree size. In this section we show that these hardness results also hold for ConDT under minimum total external path length. Our theorem relies on the observation that if I is an instance of ConDT with minimum total external path length s then I has minimum tree size at least $\Omega(\sqrt{s})$. If it didn't, a tree of smaller size would have smaller total external path length, a contradiction. The case where minimum total external path length s corresponds to minimum size $\Omega(\sqrt{s})$ is a cascading tree; that is, a tree with exactly one leaf at each depth save the deepest two.

Theorem 3. *If there exists an s^k approximation for some constant $k > 0$ to decision trees with minimum total external path length s then NP is contained in DTIME[2^{m^ϵ}] for some $\epsilon < 1$.*

Proof. Let I be an instance of ConDT with minimum total external path length $s = r^2$. It follows that I has minimum tree size at least $\Omega(r)$. Now, if an s^k approximation did exist for some k then there would exist an $\Omega(r^{2k}) = r^{k'}$ approximation for some constant k' for ConDT under minimum tree size; a contradiction. □

4 Open Problems and Discussion

Our primary result in this paper is a $(\ln n + 1)$-approximation for the decision tree problem. The most prominent open problem is the gap between the upper and lower bounds on the approximation ratio of DT. The best lower bound on the approximation ratio in the unweighted items case is $2 - \epsilon$ for any $\epsilon > 0$ (modulo P≠NP) [4]. This improves upon the no PTAS result from [10]. However, when the input has arbitrary weights on the items, then the lower bound on the approximation ratio becomes $\Omega(\log n)$.

Unfortunately, the $\Omega(\log n)$ lower bound of Laber and Nogueira [13] for decision trees of minimal height also does not apply to our problem. This is because *height* mirrors the notion of *size* in set cover problems.

Amplifying the $2-\epsilon$ gap using techniques from [9] for ConDT does not work for DT. There, one squares an instance of ConDT, applies an α-approximation, and recovers a solution to the original instance which is a $\sqrt{\alpha}$-approximation. Repeating this procedure yields the stronger lower bound. This does not work for DT because the average path length only doubles when squaring the problem, so solving the squared problem with an α-approximation and recovering a solution to the original problem simply preserves (and unfortunately does not improve) the approximation ratio. The hardness results from [1] rely on the construction of a binary function which is difficult to approximate accurately when certain instances of a hitting-set problem have large solutions. These techniques do not appear to work for DT either.

The analysis of the greedy algorithm is also not known to be tight. We only know of instances where the approximation ratio of the greedy algorithm is not better than $\Omega(\frac{\log n}{\log \log n})$ of optimal [7,12].

Finally, we leave as an open question the problem of approximating DT with *both* arbitrary item weights and arbitrary test weights.

Acknowledgments. We thank the anonymous reviewers for their insightful and helpful comments.

References

1. Alekhnovich, M., Braverman, M., Feldman, V., Klivans, A.R., Pitassi, T.: Learnability and automatizability. In: Proceedings of the 45th Annual Symposium on Foundations of Computer Science, pp. 621–630. IEEE Computer Society Press, Los Alamitos (2004)
2. Arkin, E.M., Meijer, H., Mitchell, J.S.B., Rappaport, D., Skiena, S.: Decision trees for geometric models. International Journal of Computational Geometry and Applications 8(3), 343–364 (1998)
3. Carmo, R., Donadelli, J., Kohayakawa, Y., Laber, E.S.: Searching in random partially ordered sets. Theor. Comput. Sci. 321(1), 41–57 (2004)
4. Chakaravarthy, V.T., Pandit, V., Roy, S., Awasthi, P., Mohania, M.K.: Decision trees for entity identification: approximation algorithms and hardness results. In: Libkin, L. (ed.) Proceedings of the Twenty-Sixth ACM Symposium on Principles of Database Systems, pp. 53–62 (2007)
5. Feige, U., Lovász, L., Tetali, P.: Approximating min-sum set cover. Algorithmica 40(4), 219–234 (2004)
6. Garey, M.R.: Optimal binary identification procedures. SIAM Journal on Applied Mathematics 23(2), 173–186 (1972)
7. Garey, M.R., Graham, R.L.: Performance bounds on the splitting algorithm for binary testing. Acta Inf. 3, 347–355 (1974)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
9. Hancock, T., Jiang, T., Li, M., Tromp, J.: Lower bounds on learning decision lists and trees. Information and Computation 126(2), 114–122 (1996)
10. Heeringa, B.: Improving Access to Organized Information. PhD thesis. University of Massachusetts, Amherst (2006)
11. Hyafil, L., Rivest, R.: Constructing optimal binary decision trees is np-complete. Information Processing Letters 5(1), 15–17 (1976)
12. Rao Kosaraju, S., Przytycka, T.M., Borgstrom, R.S.: On an optimal split tree problem. In: Dehne, F.K.H.A., Gupta, A., Sack, J.-R., Tamassia, R. (eds.) WADS 1999. LNCS, vol. 1663, pp. 157–168. Springer, Heidelberg (1999)
13. Laber, E.S., Nogueira, L.T.: On the hardness of the minimum height decision tree problem. Discrete Applied Mathematics 144(1-2), 209–212 (2004)
14. Moret, B.M.E.: Decision trees and diagrams. ACM Comput. Surv. 14(4), 593–623 (1982)
15. Moshkov, M.J.: Greedy algorithm of decision tree construction for real data tables. In: Transactions on Rough Sets, pp. 161–168 (2004)
16. Munagala, K., Babu, S., Motwani, R., Widom, J.: The pipelined set cover problem. In: ICDT, pp. 83–98 (2005)
17. Murthy, K.V.S.: On growing better decision trees from data. PhD thesis, The Johns Hopkins University (1996)