INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

An Optimal Constrained Pruning Strategy for Decision Trees

Hanif D. Sherali, Antoine G. Hobeika, Chawalit Jeenanunta,

To cite this article:
Hanif D. Sherali, Antoine G. Hobeika, Chawalit Jeenanunta, (2009) An Optimal Constrained Pruning Strategy for Decision Trees. INFORMS Journal on Computing 21(1):49-61. http://dx.doi.org/10.1287/ijoc.1080.0278

INFORMS Journal on Computing
Vol. 21, No. 1, Winter 2009, pp. 49–61
ISSN 1091-9856 | EISSN 1526-5528 | 09 | 2101 | 0049

DOI 10.1287/ijoc.1080.0278

An Optimal Constrained Pruning Strategy for Decision Trees

Hanif D. Sherali
Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, hanifs@vt.edu

Antoine G. Hobeika
Via Department of Civil and Environmental Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, hobeika@vt.edu

Chawalit Jeenanunta
Sirindhorn International Institute of Technology, Thammasat University, Bangkadi, Pathum Thani 12121, Thailand, chawalit@siit.tu.ac.th

This paper is concerned with the optimal constrained pruning of decision trees. We present a novel 0–1 programming model for pruning the tree to minimize some general penalty function based on the resulting leaf nodes, and show that this model possesses a totally unimodular structure that enables it to be solved as a shortest-path problem on an acyclic graph. Moreover, we prove that this problem can be solved in strongly polynomial time while incorporating an additional constraint on the number of residual leaf nodes. Furthermore, the framework of the proposed modeling approach renders it suitable to accommodate different (multiple) objective functions and side-constraints, and we identify various such modeling options that can be applied in practice. The developed methodology is illustrated using a numerical example to provide insights, and some computational results are presented to demonstrate the efficacy of solving generically constrained problems of this type. We also apply this technique to a large-scale transportation analysis and simulation system (TRANSIMS), and present related computational results using real data to exhibit the flexibility and effectiveness of the proposed approach.

Key words: decision analysis; analysis of algorithms; programming; integer; transportation
History: Accepted by Amit Basu, former Area Editor for Knowledge and Data Management; received August 2005; revised October 2006, March 2007, July 2007; accepted February 2008. Published online in *Articles in Advance* August 8, 2008.

1. Introduction and Motivation

Decision trees are popularly used in data mining scenarios for the purpose of classifying data or building regression models to extract useful information, and have been widely applied in a variety of contexts such as medical diagnosis, signal and image pattern recognition, and demographic studies in transportation planning. The construction of such trees typically follows a two-phase process: a *tree-growing step* followed by a *pruning procedure* (some exceptions are discussed below). In the tree-growing step, a training data set is progressively partitioned into nodes according to a classification scheme based on a set of designated *independent* variables that represent certain attributes of interest. (For example, in a transportation planning context where the data points represent households, the independent variables could record attributes such as the household size or the number of members in the household that fall within different age groups.) At each iterative step in this process, a leaf node of the current tree is selected and is partitioned into two subnodes based on some identified independent variable taking on values lesser than or greater than a specified *split value*. The ultimate objective is to generate a tree that classifies within each of its leaf nodes a set of data points that bear some common resemblance with respect to certain *dependent* variables that measure relevant performance characteristics. (In the aforementioned transportation planning context, the dependent variables could represent the number of trips made by the household or the time spent performing different activities over a 24-hour horizon.) This is achieved by attempting to minimize some classification *error function*, as for example, the (weighted) sum of squared differences between each dependent variable value and its mean value among the data points residing within each leaf node. (Equation (5) in §3 provides a formal mathematical definition.) Hence, each partitioning step is executed in a manner such that the resulting decrement in this error function is maximized. This process continues until all

the current leaf nodes are sufficiently refined, whence they are declared to be *terminal nodes*.

The resulting tree purposefully overfits the data. Consequently, a subsequent pruning strategy is applied to collapse the tree to a desired extent. The focus in this paper is on conducting an optimal constrained pruning of decision trees. In general, such trees might be *classification trees*, where the dependent variables are categorical, or might be *regression trees*, where the dependent variables are numerical and where one might be further interested in developing a functional mapping from the independent to the dependent variables to enable a prediction of the latter in terms of the former (see Breiman et al. 1984, Murthy 1998 for a treatise on decision trees).

A prominent decision tree development procedure that has been widely applied in practice is the classification and regression tree (CART) algorithm, which is described in Breiman et al. (1984). Given a tree T that is grown via the foregoing procedure, the CART algorithm recommends two alternative schemes for reducing or *pruning* this tree, namely, the *cross-validation approach* and the *independent test sample approach*. For small sample sizes, it is recommended to use the cross-validation approach, in which some 10 different trees are usually grown, each built from a different 90% of the total sample data set. The prescribed tree is then selected as the one that yields the lowest prediction error or misclassification cost based on the entire data set. However, when the sample size is very large, Breiman et al. (1984) recommend the less computationally intensive independent test sample approach. This method adopts the popular *cost-complexity pruning* (CCP) technique, which repeatedly determines the weakest nonterminal node junction of the tree in the sense that such a node yields the smallest value of a defined *penalty function* if the subtree rooted at this node is collapsed into it. Performing this collapsing step repeatedly produces a sequence of nested trees leading to the single root node of the original tree. In this sequence of nested subtrees, each subtree is optimal in the sense that it has the least total cost among all possible subtrees having the same number of leaf nodes (i.e., having the same *complexity*). The optimal tree is prescribed as the one that yields the lowest prediction error among the resulting subtrees based on classifying another independent test sample using each of these subtrees. Breiman et al. (1984) and Breslow and Aha (1997) describe various other (mostly heuristic) pruning methods that can be used in this context, but by far, the most widely used technique is the aforementioned CCP method, which has been implemented in many other systems such as S-plus and OC1 as well (see Breiman et al. 1984).

Among several alternative criteria used for pruning decision trees, one that has been extensively analyzed is the *minimum description length* (MDL) principle, which seeks to minimize the bits required to encode the tree itself as well as to encode the classes of records that reside in the leaf nodes. Quinlan and Rivest (1989) discuss how to construct optimal decision trees based on this criterion, and Mehta et al. (1995) propose an iterative bottom-up pruning procedure that is governed by this principle. Note that using the approach presented in this paper, we can minimize a general weighted sum of the encoding lengths for the tree and for the records classified within the leaf nodes, in addition to incorporating alternative objective criteria and constraints. Other examples of objective criteria that are used in data mining contexts, which we can accommodate as well, are the error rate and the standard error (see Ye 2003). Here, based on a scheme for identifying a data point as being erroneously misclassified within a given set of points, the error rate (ER) is defined as the fraction of data points that are in error, and the standard error is defined as $\sqrt{\mathrm{ER}(1-\mathrm{ER})/d}$, where d is the total number of data points. Some additional objective criteria, as motivated by our transportation planning case study, are discussed in detail in §3, along with related accompanying side-constraints.

Li et al. (2001) propose an interesting alternative *dynamic-programming-based pruning* (DPP) algorithm. For each possible tree size defined in terms of the number of leaf nodes, this routine recursively determines a corresponding optimal pruned tree that would have a minimal prediction error. Consequently, the pruned trees produced by the CCP method are a subset of the trees produced over the different recursive stages of the DPP technique, where each stage is defined based on the number of leaf nodes, and where this number is decremented by a unit at each algorithmic step. Hence, the CCP method does not determine optimal solutions for each designated number of leaf nodes. Moreover, in the computational results presented in Li et al. (2001), the DPP algorithm is shown to perform better than the CCP procedure in terms of classification accuracy. It is also worth noting that the sequence of pruned trees produced by the DPP approach are not necessarily nested, nor are they unique, but as Li et al. (2001) argue, this issue is not of concern in regard to the quality of these trees.

Our focus in this paper is also on the pruning step, given that an (overfitted) decision tree has already been generated via an appropriate tree-growing process. However, in addition to minimizing some defined penalty function over all the resulting leaf nodes, we might be interested in also optimizing certain secondary objective functions (or some combinations thereof) that examine other statistics related to the classified data. Furthermore, we might be interested in imposing additional side-constraints governing alternative objective goals or related to the size,

statistical properties, or structure of the resulting tree. (Specific examples of such alternative objective functions and constraints are discussed in §3.) In this sense, the DPP and CCP methods as originally presented aim to minimize the misclassification error subject to a constraint on the total number of leaf nodes (where, as mentioned above, CCP solves this problem only for a subset of possible restrictions on the number of leaf nodes). To modify these methods to handle more complex optimal constrained pruning problems would require a customized revision of the underlying methodology. For example, to accommodate additional constraints in the DPP approach, we would need to appropriately redefine the dynamic programming state space in terms of a multidimensional vector having components corresponding to each of the constraints, and restructure the accompanying recursive equation (see Bertsekas 2001). On the other hand, the mathematical programming framework developed in this paper can *automatically* accommodate arbitrary side-constraints and also analyze multiple objectives as necessary.

In the context of constrained pruning, Chou et al. (1989) examine a tree-structured vector quantization design problem for which they propose a heuristic pruning procedure for minimizing the expected distortion subject to a single cost constraint. Lin et al. (1992) have subsequently analyzed the complexity of solving such problems. Using a dynamic programming framework similar to that of Li et al. (2001), they establish that when the single constraint simply restricts the number of final leaf nodes generated, then the underlying problem can be solved in polynomial time. On the other hand, under more intricate single constraints that limit the entropy or the expected depth of the leaf nodes, they demonstrate that the pruning problem becomes NP-hard. However, no solution approaches are delineated for the latter problems. In this paper, using an alternative 0–1 modeling approach, we not only establish the same polynomial-time complexity for the problem involving the minimizing of a general misclassification or distortion or encoding description function subject to a constraint on the number of leaf nodes, but also, we show how this model provides the facility to conduct an optimal pruning subject to a variety of multiple constraints.

The main contribution of this research, therefore, is to develop a new concept of formulating a specially structured 0–1 programming model for optimally pruning decision trees based on specified objective functions and restrictions that govern certain structural and statistical properties of the generated classification tree. We demonstrate that the basic underlying formulation possesses a totally unimodular structure that enables it to be solved as an equivalent shortest-path problem on an acyclic graph. Furthermore, we show that this framework also permits solving the model with an additional constraint on the number of leaf nodes in strongly polynomial time. For problems having arbitrary side-constraints, the basic problem structure greatly facilitates its solution, as demonstrated by our computational results. Finally, we apply this general concept within the context of a large-scale *transportation analysis and simulation system* (TRANSIMS) (see Los Alamos National Laboratory 2002a).

Since the writing of this paper, an associate editor brought to our attention a manuscript by Zhang and Huei-chuen (2005), wherein an alternative 0–1 programming model is presented for pruning decision trees to minimize an error function with respect to a test set (without additional constraints). Their model also possesses a totally unimodular structure as does ours, although our model is more compact in representation (see Remark 1 in §2 for a more detailed comparison). However, Zhang and Huei-chuen (2005) do not discuss the incorporation of additional side-constraints or provide any implementation results; rather, their focus is on presenting an alternative complexity proof of the bottom-up pruning method due to Quinlan (1988).

Before proceeding, it is important to emphasize that our present focus is on the pruning step and not on the tree-growing mechanism, although the latter naturally governs the final tree produced. The motivation here is that the primary objective utilized in our proposed modeling approach is to minimize the misclassification error as addressed in the tree-growing procedure, but along with the consideration of other secondary objectives, in addition to certain imposed side-constraints. Nonetheless, the joint consideration of tree growing and pruning in an optimization framework deserves further research. In this connection, Rastogi and Shim (1998) describe a pruning method, called PUBLIC, in which tree building and pruning are integrated by using a lower bound to determine a priori whether the subtree rooted at a given node will be later pruned, and thereby obviating the expansion of such a node. The specific criterion for which such a lower bound is derived is the MDL objective discussed above. However, additional restrictions such as limiting the number of leaf nodes cannot be automatically integrated within this scheme. Nonetheless, their concept for jointly considering the tree-growing and pruning steps is novel and deserves further research.

Adopting an alternative framework, Folino et al. (1999, 2001, 2002) investigate the design of both serial and parallel implementations of genetic algorithms for heuristically evolving a population of trees with

respect to certain appropriately defined fitness functions. Also, Fu et al. (2003a, b, 2006) propose certain novel modeling concepts and genetic algorithmic approaches for constructing decision trees based on very large data sets.

In another optimization-related tree construction method, Bennett (1992) proposes a linear programming model for determining linear-combination splits at nodes for two-class decision trees. Also, Bennett (1994) has studied how splitting decisions should be made at each level of a given particular final tree structure so that the resultant classification error with respect to two classes of data points would be minimized. The splits at each level are made using general linear functions, and the problem is formulated as a nonconvex multilinear program, which is solved approximately using the Frank-Wolfe algorithm. Bredensteiner and Bennett (1998) study a related problem of splitting data comprised of two classes of points using a separating hyperplane such that the number of nonzero hyperplane coefficients (which represents the number of features or attributes used) is minimized, given a specified misclassification error tolerance. The problem is formulated as a parametric bilinear program, which is NP-hard, and a heuristic based on the Frank-Wolfe method is adopted to solve this problem. For further reading and surveys on constructing decision trees, we refer the reader to Breslow and Aha (1997), Li et al. (2001), Murthy (1997, 1998), and Safavin and Landgrebe (1991).

The remainder of this paper is organized as follows. In §2, we develop our basic 0–1 programming model for optimally pruning decision trees, and we analyze its structure and establish its solvability with or without an additional constraint on the number of leaf nodes in strongly polynomial time. For more generally constrained problems, we provide some computational experience to demonstrate the efficacy of using our generic modeling approach. Section 3 discusses the potential use of this optimization model to conduct an optimal pruning based on specified (possibly multiple) objective functions and subject to desired side-constraints, and §4 illustrates this approach using a numerical example. Section 5 describes a real-life transportation planning application and presents related computational results. Finally, §6 closes this paper with a summary and conclusions.

2. Optimal Constrained Pruning Strategy

This section describes a 0–1 programming approach for optimally pruning a given tree T based on various specified objectives, goals, and side-constraints. To present the proposed methodology, let us begin by introducing some notation and terminology.

Let T be the binary tree obtained via some tree-growing procedure, and let $\widetilde{T}$ represent its terminal nodes, with cardinality $|\widetilde{T}| \equiv m$. For convenience in notation, we will index the nodes of T as $h = 0, 1, \ldots, n$, where node 0 represents the root node, and denote this set of nodes by $\mathcal{N} = \{0, 1, \ldots, n\}$. For any $h \in \mathcal{N}$, let us define the following (standard) terminology (see Bazaraa et al. 2005, for example):

$p(h) =$ set of *predecessor nodes* of $h \equiv$ set of nodes (excluding h) that lie on the chain from h to the root node 0 ($p(0) = \varnothing$).

$p_I(h) =$ *immediate predecessor* of node $h \equiv$ first node adjacent to h that lies on the chain from h to the root node.

$T_h =$ subtree of T that is *rooted at node* h (including h). (Hence, $T_0 \equiv T$.)

$b(h) =$ *brother* of node h (exists for $h \neq 0$) $=$ node for which $p_I[b(h)]$ equals $p_I(h)$. (Note that $b[b(h)] = h$ because T is a binary tree.)

$s(h) =$ set of *successor nodes* of $h \equiv$ set of nodes in $T_h - \{h\} \equiv$ set of nodes u for which $h \in p(u)$.

$s_I(h) =$ the two *immediate successors* (or *children*) of node h (if they exist), which are produced by partitioning node h within the tree-growing procedure.

Furthermore, for the sake of convenience, we will henceforth refer to the (unique) chain from any node u to a node v in T, including the nodes u and v, by $C[u \to v]$. When we wish to exclude nodes u, or v, or both from this chain, we will use the notation $C(u \to v]$, $C[u \to v)$, and $C(u \to v)$, respectively. Finally, the process of pruning a subtree T_h that is rooted at h by absorbing it within the node h will be referred to as *collapsing* T_h *into* h. Likewise, any node that is a part of such a subtree T_h (excluding the node h) will be called a *collapsed node*.

Any legitimate pruning of the tree T will comprise of collapsing entire subtrees rooted at particular nodes into these corresponding nodes, thereby producing a resultant *pruned tree*, $T_{\text{Prune}}(E)$, having E as its set of leaf (or end) nodes. A formal definition of a pruned tree is given below.

Definition 1. We will say that $T_{\text{Prune}}(E)$ is a pruned tree of T having leaf nodes E iff:

(i) $T_{\text{Prune}}(E)$ is a subtree of T that contains the root node 0, has $E \subseteq \mathcal{N}$ as its set of leaf nodes, and is given by $\bigcup_{h \in E} C[h \to 0]$, and

(ii) for any node $h \neq 0$ that is included in $T_{\text{Prune}}(E)$, we also have that its brother node $b(h)$ is included within $T_{\text{Prune}}(E)$.

Observe that we require condition (ii) above to ensure, along with condition (i), that any collapsed node is part of an entire collapsed subtree that is rooted at some node. In other words, (ii) equivalently asserts that $T_{\text{Prune}}(E)$ is a *binary tree*.

Now, let us define a set of binary variables as follows:

$$z_h = \begin{cases} 1 & \text{if node } h \text{ is selected as a leaf node,} \\ 0 & \text{otherwise,} \quad \forall\, h \in \mathcal{N}, \end{cases} \tag{1}$$

and consider the following system of constraints:

$$z_h + \sum_{u \in p(h)} z_u = 1 \quad \forall\, h \in \widetilde{T}, \tag{2a}$$

$$z_h \text{ binary} \quad \forall\, h \in \mathcal{N}. \tag{2b}$$

Proposition 1 asserts that system (2) precisely characterizes the set of pruned trees of T in the sense of Definition 1.

Proposition 1. *Let $T_{\text{Prune}}(E)$ be a pruned tree of T having leaf nodes E. Define a (binary) vector $z \in \{0,1\}^{n+1}$ having components $z_h = 1$ if $h \in E$, and $z_h = 0$ otherwise. Then, z satisfies system* (2). *Conversely, consider any feasible solution z to system* (2). *Define*

$$E \equiv \{h\colon z_h = 1\} \tag{3a}$$

and let τ be the subtree of T given by

$$\tau \equiv \bigcup_{h \in E} C[h \to 0]. \tag{3b}$$

Then, τ is a pruned tree of T having leaf nodes E; i.e., $\tau \equiv T_{\text{Prune}}(E)$.

Proof. See the appendix. □

The next result establishes a critical structural property of system (2).

Proposition 2. *The constraint matrix of system* (2a) *is totally unimodular.*

Proof. It is sufficient to show that a permutation of the rows of (2a) yields an *interval coefficient matrix* (see Nemhauser and Wolsey 1999); i.e., each column has a set of consecutive ones as its nonzero elements. To see this, arrange the nodes of T in a *preorder list structure* P (see Bazaraa et al. 2005); i.e., for each node $u \in \mathcal{N}$, the set of successor nodes $s(u)$ (and hence, the subtree rooted at node u) immediately follows u in the list. Suppose now that we write the constraints (2a) with $h \in \widetilde{T}$ selected in order of its appearance within P. Then, each variable z_h, for $h \in \widetilde{T}$, has a unit vector column, and each variable z_u, for $u \notin \widetilde{T}$, appears only in the set of consecutive constraints written for $h \in \widetilde{T} \cap s(u)$, where $\widetilde{T} \cap s(u)$ are the leaf nodes of the subtree of T that is rooted at u. This completes the proof. □

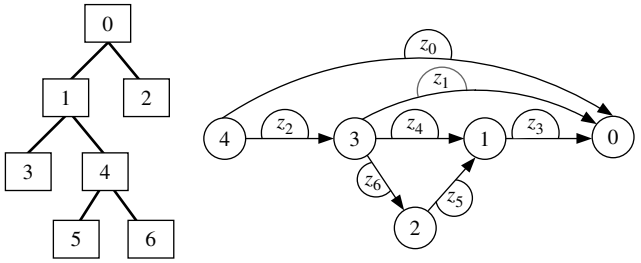
By Propositions 1 and 2, system (2) provides a mathematical structure that affords a great deal of flexibility in designing an optimal constrained pruning strategy. Noting that any feasible solution to (2) maps onto a pruned tree $T_{\text{Prune}}(E)$ as defined in Proposition 1 via (3), we can ascribe any desired *penalty function* $f(h)$ to the leaf nodes of the prescribed pruned tree (e.g., Breiman et al. 1984 describe one such penalty function used with the CCP technique that is based on a combination of the classification error and the number of residual leaf nodes), and choose to solve the *pruning problem*

$$\text{PP1: Minimize}\left\{\sum_{h=0}^{n} f(h) z_h \text{: system (2)}\right\}. \tag{4}$$

Note that for any $h \in \mathcal{N}$, if h turns out to be a leaf node of $T_{\text{Prune}}(E)$ (i.e., $h \in E$) as indicated by $z_h = 1$, then by examining the restrictions on the branches in the chain $C[h \to 0]$ in T, we know precisely the classification characteristics of node h and what data points would be classified to reside within node h. Moreover, because any solution to (2) produces a legitimate pruned tree, it would result in a unique classification of data into the leaf nodes $h \in E$ given by (3a). Note that we can consider a variety of penalty functions, possibly in combination as multiple (preemptive) objective functions, for formulating (4) as described in §3. Moreover, by Proposition 2, (4) has a structure that permits it to be solved in strongly polynomial time as a special shortest-path problem on an acyclic graph, even with an additional constraint on the total number of leaf nodes that are permitted in the final pruned tree. This is formally established and illustrated subsequently below.

Proposition 3. *Problem* PP1 *given by* (4) *can be recast as a shortest-path problem on an acyclic graph having $m+1$ nodes and $n+1$ arcs, where $m \equiv |\widetilde{T}|$ and $\mathcal{N} = \{0, 1, \ldots, n\}$, and can thereby be solved in $O(n)$ time.*

Proof. Consider an ordering of the $|\widetilde{T}| = m$ constraints in (2a) according to the preorder list P as in the proof of Proposition 2, so that the coefficient matrix displays an interval matrix structure. Label these rows as $R_1, R_2, \ldots, R_m$. Now, let us rewrite these constraints in the following equivalent form using the indicated row operations: $\{-R_1, R_1 - R_2, R_2 - R_3, \ldots, R_{m-1} - R_m, R_m\}$, where we have accommodated a single redundant row for convenience (observe that the sum of these rows equals a null equation). By the interval matrix structure (see Nemhauser and Wolsey 1999), this resultant equation system displays a network-flow structure on a graph G, having $m+1$ nodes, one associated with each row in the transformed system, and having $n+1$ arcs, one associated with each of the variables z_h, $h = 0, 1, \ldots, n$. Furthermore, numbering the nodes of this graph G as $\{0, 1, \ldots, m\}$ according to the transformed equations, and noting that the right-hand sides of the transformed equations are respectively given by



(a) Classification tree (b) Acyclic graph G for the shortest-path problem

Figure 1 Decision Tree and Acyclic Graph G for Illustrating Proposition 3

$\{-1, 0, \ldots, 0, 1\}$, we have that node m has a supply of 1, node 0 has a demand of 1 (supply of -1), and the remaining nodes are transshipment nodes in the transformed problem (see Bazaraa et al. 2005, for example, for this standard network representation). Moreover, by the nature of the transformation, each column of z_h, $h \in \mathcal{N}$, has a -1 and $+1$ entry, appearing in that order in the transformed rows. Hence, each arc begins at a node that has a higher index than the node at which it ends. Therefore, the graph is acyclic. Consequently, the transformed problem seeks a shortest path from node m to node 0 on this resulting acyclic graph, and displays a *topological ordering* of nodes $\{m, \ldots, 1, 0\}$ (i.e., arcs go from earlier nodes to later nodes in this ordered list). By Ahuja et al. (1993) (see Theorem 4.3, p. 108), this problem can be solved in $O(n)$ time. □

To illustrate the construction in the proof of Proposition 3, consider the decision tree depicted in Figure 1(a), having $n = 6$ and $m = 4$. The set of leaf nodes is given by $\widetilde{T} = \{3, 5, 6, 2\}$, where these nodes are arranged according to the preorder list P as in the proof of Proposition 2. The corresponding system (2) with the rows $R_1, \ldots, R_4$ labeled as in the proof of Proposition 3, and the resulting equivalent system after performing the designated row operations, are given in Figure 2. The latter system represents the constraints for the shortest-path problem on the acyclic graph G depicted in Figure 1(b), where the $(m + 1) = 5$ nodes of G are associated with the

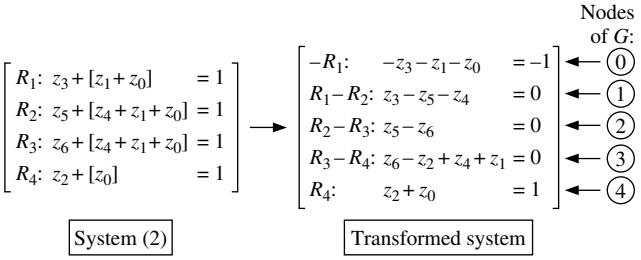


Figure 2 System (2) and Its Transformed Form for the Decision Tree of Figure 1

rows of the transformed system as displayed in Figure 2, and the $(n + 1) = 7$ arcs are associated with the columns of the variables $z_0, z_1, \ldots, z_6$ in this system.

Proposition 4. *Given any penalty function $f(h)$, $h \in \mathcal{N}$, Problem* PP1 *with an additional constraint on the number of leaf nodes of the type $\sum_{h\in\mathcal{N}} z_h \geq E_{\min}$, or $\sum_{h\in\mathcal{N}} z_h \leq E_{\max}$, or $\sum_{h\in\mathcal{N}} z_h = E_{\max}$, for some parameters $E_{\min}$ and $E_{\max} \in \{1, \ldots, m\}$, can be solved in $O(mn)$ time.*

Proof. For an arbitrary penalty function $f(h)$, $h \in \mathcal{N}$, Sherali (1991) presents an algorithm for the shortest-path problem identified in (the proof of) Proposition 3 that determines the best q-step path from node m to node 0 for all values of $1 \leq q \leq m$ in (strongly) polynomial time of overall complexity $O(mn)$. Noting that the best q-step path for each such value of $q \in \{1, \ldots, m\}$ corresponds to an optimal solution to Problem PP1 with the additional constraint that precisely q of the z-variables equal one, we can therefore solve (4) with the additional stated constraints in the proposition in $O(mn)$ time. □

In the following section, we shall discuss different possible objective functions and side-constraints that can be accommodated within Problem PP1 in a general strategic pruning approach. Note that even in cases when we might incorporate additional side-constraints that might no longer preserve a network-flow structure, the resultant problem still has a special totally unimodular set-partitioning substructure that promotes its solvability (see Nemhauser and Wolsey 1999). In general, such a 0–1 integer program with a few side-constraints could be solved to optimality quite effectively using a standard package such as CPLEX (version 9.0; 2005), although in theory, the underlying problem with arbitrary side-constraints becomes NP-hard. To illustrate the efficacy of solving Problem PP1 under such arbitrary additional side-constraints that augment system (2), we conducted the following experiment.

We generated a test-bed of binary decision trees T by randomly selecting the number of nodes at each depth or level of the tree that are declared to be leaf nodes of T. In this scheme, if $l = 0, 1, \ldots, L$ denotes the levels of the constructed binary tree T, and n_l and e_l, respectively, represent the number of nodes and the number of leaf nodes at level l, we begin with $n_0 = 1$ and $e_0 = 0$, and then recursively, given n_{l-1} and $e_{l-1} < n_{l-1}$ for $l \in \{1, \ldots, L\}$, we set $n_l = 2(n_{l-1} - e_{l-1})$, and randomly generate $e_l \in \{0, \ldots, n_l - 1\}$, except that we set $e_L = n_L$. Hence, $m \equiv |\widetilde{T}| = \sum_{l=1}^{L} e_l$ and $n \equiv |\mathcal{N}| - 1 = \sum_{l=1}^{L} n_l$. The function values $f(h)$, $\forall\, h \in \mathcal{N} \equiv \{0, 1, \ldots, n\}$ were generated randomly to be integers in the interval $[1, 25]$. The resultant Problem PP1 was cast in the transformed format of a shortest-path problem on an acyclic graph G having $m + 1$ nodes and $n + 1$ arcs as described in the proof of Proposition 3,

Table 1 Computational Effort in CPU Seconds for Solving Problem PP1 with Arbitrary Side-Constraints

				(K, t)					
Problem	$\|\tilde{T}\| = m$	$\|\mathcal{N}\| = n+1$	$K=0$	(1, 1)	(1, 2)	(3, 1)	(3, 2)	(5, 1)	(5, 2)
1	51	101	0	0	0	0	0	0	0
2	64	127	0	0	0	0	0	0	0
3	72	143	0	0	0	0	0	0	0
4	82	163	0	0	0	0	0	0	0
5	89	177	0	0	0	0	0	0	0
6	101	201	0	0	0	0	0	0	0
7	108	215	0.01	0.01	0.01	0.01	0.01	0.01	0.01
8	111	221	0.01	0.01	0.01	0.01	0.01	0.01	0.01
9	129	257	0.01	0.01	0.01	0.01	0.01	0.01	0.01
10	260	519	0.01	0.01	0.01	0.01	0.01	0.01	0.01
11	299	597	0.02	0.02	0.02	0.02	0.02	0.02	0.02
12	327	653	0.02	0.02	0.02	0.02	0.02	0.02	0.02

and to this problem, we appended some $K \in \{1, 3, 5\}$ arbitrary side-constraints of the form $\sum_{h=0}^{n} \alpha_{kh} z_h \le \beta_k$, $k = 1, \ldots, K$, where the α_{kh}-coefficients were randomly generated so that $\alpha_{kh} \in \{0, 1\}$ for problems of type $t = 1$ and $\alpha_{kh} \in \{0, 1, 2, 3\}$ for problems of type $t = 2$, and where β_k, $\forall k = 1, \ldots, K$ were uniformly generated as integers in the intervals $[1, 5]$ and $[1, 12]$ for $t = 1$ and 2, respectively. The problems were solved using the commercial package CPLEX (2005), version 9.0, on a Dell Precision 650 workstation having a Xeon™ CPU 2.40 GHz and 1.50 GB of RAM. Table 1 presents the results obtained, including for the case $K = 0$ (no side-constraints). Note that all problems ranging in size to about 500 nodes and 1,000 arcs were solved within 0.02 CPU seconds.

Remark 1. In concluding this section, we comment further on the alternative 0–1 integer programming (IP) model of Zhang and Huei-chuen (2005) that was mentioned in §1. In contrast with system (2), which compactly captures the required characteristics delineated in Definition 1 of a pruned tree as established by Proposition 1 for formulating Problem PP1, Zhang and Huei-chuen (2005) explicitly model each such property identified in Definition 1 using additional variables and constraints. As a result, their IP model has $2(n + 1)$ variables and $3(n + 1) - m + 1$ constraints (in contrast with $n + 1$ variables and m constraints for our model, where m is typically much smaller than n). Furthermore, although their model without additional constraints also displays a total unimodular structure, we are able to show via our model formulation that the underlying pruning problem can be solved in polynomial time even with an additional side-constraint on the number of leaf nodes, as established in Proposition 4. In addition, because our model explicitly tracks the leaf nodes of the pruned tree, which is an important characteristic of the tree, it facilitates the adoption of different objective functions and side-constraints as delineated in §3. This is elusive in the model due to Zhang and Huei-chuen (2005), where the binary decision variables are defined to simply indicate whether a node belongs to the final tree or not, which would therefore require additional model manipulations to identify the leaf nodes. Besides, the focus in Zhang and Huei-chuen (2005) is to use their IP formulation to establish a new optimality proof of the bottom-up pruning method due to Quinlan (1988), as opposed to actually modeling and solving the types of constrained pruning problems addressed herein. □

3. Alternative Objective Functions and Side-Constraints

In §1, we mentioned several objective functions and constraints that arise within the context of machine learning and data mining applications of decision trees. In this section, we describe some additional objective functions and constraints that can be utilized in concert with the proposed optimal pruning problem formulation, which are, in particular, motivated by a practical transportation planning application. In §5, we provide computational results utilizing such objective and constraint functions in the context of this application.

3.1. Objective Functions

A common objective function used with decision trees concerns minimizing an appropriate misclassification error measure. For example, suppose that we have a training data set $D \equiv \{1, \ldots, d\}$ comprised of d observations, and that we are interested in conducting a classification based on some p performance characteristics (dependent variables) Y_j, $j = 1, \ldots, p$. Let Y_{ij} denote the value of the variable Y_j for the ith observation $\forall i = 1, \ldots, d$, $j = 1, \ldots, p$. Furthermore, let s_j denote an estimate of 1/[variance of Y_j] (e.g., as derived from Y_{ij}, $i = 1, \ldots, d$) $\forall j = 1, \ldots, p$. Then, denoting by D_h the subset of D that would be partitioned into node h were it to become a leaf node of the pruned tree, we can define a misclassification *error function* as follows:

$$f_1(h) \equiv \sum_{j=1}^{p} \sum_{i \in D_h} s_j \left(Y_{ij} - \overline{Y}_{D_h j}\right)^2 \quad \forall h \in \mathcal{N},$$

$$\text{where} \quad \overline{Y}_{D_h j} \equiv \frac{\sum_{i \in D_h} Y_{ij}}{|D_h|}. \quad (5)$$

For example, in the transportation planning application discussed in §5 (and alluded in §1), the training data set D pertains to survey households, and the variables Y_j, $j = 1, \ldots, p$ represent performance characteristics of the households with respect to activities such as the total number of trips made or the total amount of time spent at home or work, etc., over a

24-hour horizon. The use of (5) then attempts to classify households having similar characteristics within the terminal nodes of the tree.

To compose a suitable objective function for Problem PP1, we could impose a constant penalty $\alpha > 0$ for each leaf node created by defining

$$f_2(h) \equiv \alpha, \tag{6a}$$

so that $\sum_h f_2(h)z_h \equiv \alpha \sum_h z_h$. Then, as in the CCP method, we could adopt $f(h) = f_1(h) + f_2(h)$ within (4) for a suitable penalty parameter $\alpha > 0$. Alternatively, in lieu of using a constant α for each node h as in (6a), we can use a factor

$$\alpha_h \equiv \alpha \cdot (\text{depth of node } h), \tag{6b}$$

where the *depth* of a node h is the number of arcs on the chain $C[h \to 0]$. Then, by defining the function

$$f_3(h) \equiv \alpha_h \quad \forall\, h \in \mathcal{N}, \tag{6c}$$

we could likewise use this in combination with $f_1(h)$ to yield $f(h) = f_1(h) + f_3(h)$, which would tend to produce relatively more breadthwise expanded trees. In this same spirit, we can choose to minimize the maximum depth over the leaf nodes subject to additional constraints that restrict the minimal resulting tree size by solving

$$\text{PP2: Minimize}\{\eta:\ \eta \ge f_3(h)z_h \quad \forall h \in \mathcal{N},\ \text{system (2)},\ \text{plus additional constraints}\}. \tag{7}$$

Another possible objective function can be based on some particular important characteristic Y_{j^*}, $j^* \in \{1, \ldots, p\}$, which we might be interested in maximizing with respect to a given synthetic data set D' when it is assigned to the leaf nodes of the pruned tree. In our transportation planning application, the total number of trips over a day is one such characteristic, and we might be interested in maximizing the minimum total value of this measure over the leaf nodes, subject to the restriction that we create at least some $E_{\min}$ leaf nodes. Defining D'_h as the subset of D' that would be ascribed to node h, given that node h becomes a leaf node of the prescribed pruned tree, and letting Y'_{ij^*} represent the value of the attribute Y_{j^*} for each data point $i \in D'$, we can construct the merit function

$$f_4(h) \equiv \sum_{i \in D'_h} Y'_{ij^*}, \quad \forall\, h \in \mathcal{N}, \tag{8}$$

and then solve the problem

$$\text{PP3: Maximize}\left\{\eta:\ \eta \le f_4(h)z_h\ \forall\, h \in \mathcal{N},\ \sum_{h \in \mathcal{N}} z_h \ge E_{\min},\ \text{and system (2)}\right\}. \tag{9}$$

In a likewise fashion, suppose that a decision tree is developed using a training data set D for the purpose that it will be subsequently used to classify another synthetic data set D', where the elements of D' that fall into each leaf node will be matched with elements of D that reside in the same leaf node to generate some relevant information (e.g., travel activity patterns in the context of transportation planning) for the synthetic data elements. In this regard, we might not want to have too many data elements from D' being matched with relatively fewer elements in D within any leaf node so as to avoid excessive duplications of generated information (e.g., too many similar activity patterns in the aforementioned transportation planning example). Therefore, defining D_h and D'_h as above, we might wish to minimize the sum over all the leaf nodes of the ratios of the number of synthetic versus training data observations that fall into the same leaf node, leading to

$$f_5(h) \equiv \frac{|D'_h|}{|D_h|} \quad \forall\, h \in \mathcal{N}. \tag{10}$$

Other side-constraints on the number of leaf nodes can be imposed along with (2) when using the objective to minimize $\sum_{h \in \mathcal{N}} f_5(h)z_h$, to obviate simply collapsing the tree to the root node. Alternatively, we can also minimize the maximum of these ratios at the resultant leaf nodes by solving the following modified pruning problem (for some parameter $E_{\min}$):

$$\text{PP4: Minimize}\left\{\eta:\ \eta \ge f_5(h)z_h\ \forall\, h \in \mathcal{N},\ \sum_{h \in \mathcal{N}} z_h \ge E_{\min},\ \text{and system (2)}\right\}. \tag{11}$$

We can also handle multiple objectives via a suitable goal-constrained or weighting scheme. For example, letting D^* be the total misclassification error of the original tree T, we could compose a pruned tree that minimizes the objective function characterized by f_5, say, as given by (10), while requiring the error of the resulting pruned tree to increase by no more than some permissible tolerance $100\Delta\%$ of the original error δ^*, say, where $\Delta \ge 0$. This can be achieved by solving the following pruning problem:

$$\text{PP5: Minimize}\left\{\sum_{h=0}^{n} f_5(h)z_h:\ \sum_{h=0}^{n} f_1(h)z_h \le \delta^*(1+\Delta),\ \text{along with system (2)}\right\}. \tag{12}$$

In the same spirit, multiple objective functions that are ordered by some preemptive priority scheme can be handled via an equivalent nonpreemptive weighted sum of objective functions, with weights

computed as prescribed in Sherali and Soyster (1983). Alternatively, we can sequentially minimize these functions in turn while imposing constraints (similar to (12)) that restrict each previously considered objective function to take on values no greater than (or simply equal to) the corresponding optimal value found when this function was treated (in turn) as the model's objective function. We illustrate both these options in §4.

3.2. Additional Side-Constraints

Within the context of Problem PP1, or problems of the type PP2, PP3, or PP4 given, respectively, by (2), (7), (9), and (11), we could impose various additional suitable restrictions. For example, to restrict the number of leaf nodes to control the size of the resulting tree, we could include the constraint

$$\sum_{h\in\mathcal{N}} z_h \geq E_{\min}, \quad \text{or} \quad \sum_{h\in\mathcal{N}} z_h \leq E_{\max}, \quad \text{or} \quad \sum_{h\in\mathcal{N}} z_h = E_{\max}, \tag{13}$$

as addressed in Proposition 4. For example, when solving Problem PP3 defined by (9), the role played by the restriction $\sum_{h\in\mathcal{N}} z_h \geq E_{\min}$ is to preserve a tree having at least a certain size with respect to its terminal nodes, where omitting this constraint from Problem PP3 would result in the tree T being collapsed to a single node. Likewise, when minimizing the error measure defined by (5), we would like to force the tree to be pruned to a specified maximum size imposed by a side-constraint of the type $\sum_{h\in\mathcal{N}} z_h \leq E_{\max}$. Observe that the DPP procedure discussed in §1 determines optimal solutions to the particular Problem PP1 with the additional constraint $\sum_{h\in\mathcal{N}} z_h = E_{\max}$ simultaneously for all possible values of $E_{\max}$, and the CCP method solves this problem only for some subset of such values (see Li et al. 2001 and the illustrative example given in §4). On the other hand, our modeling approach directly determines an optimal solution for any of the foregoing constraints having specified particular parameter values. Moreover, we can more generally accommodate an arbitrary set of some K constraints

$$\sum_{h\in\mathcal{N}} \alpha_{kh} z_h \leq \beta_k \quad \text{for } k = 1, \ldots, K, \tag{14}$$

for some suitable parameters α_{kh} and β_k $\forall k = 1, \ldots, K$, $h \in \mathcal{N}$. For example, one such constraint $k \in \{1, \ldots, K\}$ might restrict the average depth of the leaf nodes, given by using (6b) with $\alpha \equiv 1$ as

$$\left[\sum_{h\in\mathcal{N}} \alpha_h z_h\right] \Big/ \left[\sum_{h\in\mathcal{N}} z_h\right] \leq \mu \tag{15}$$

for some specified parameter μ. Hence, in this case, we would have $\alpha_{kh} = (\alpha_h - \mu)$ $\forall h \in N$, and $\beta_k \equiv 0$ in (14).

Likewise, with the same motivation as for (10) and (11), we can optimize some desired objective function while restricting the values of the ratios identified in (10) at leaf nodes by

$$\text{setting } z_h = 0 \quad \text{whenever } \frac{|D'_h|}{|D_h|} > R^* \quad \text{for any } h \in \mathcal{N}, \tag{16}$$

where R^* is some suitable positive parameter. Sometimes, we might like to preserve certain leaf nodes $E_1 \subseteq \widetilde{T}$. For example, there might be leaf nodes of T that classify either senior citizens, or minors, or some peculiar demographic characteristic of interest such as households having only two workers and a total income of no more than \$50,000 in a transportation planning context, which might likely be collapsed when solving a problem of type (12), for example. However, because of the special nature of these leaf nodes from a statistical record point of view, we might prefer to preserve such nodes in the pruned tree. In such a case, we can simply impose the restriction

$$z_h = 1, \quad \forall h \in E_1. \tag{17}$$

Likewise, we might wish to enforce that certain nodes E_0 of T do *not* turn out to be leaf nodes (i.e., they are either collapsed, or are preserved as nonterminal nodes of the prescribed pruned tree). This can be accommodated by setting $z_h = 0$ $\forall h \in E_0$. Equation (16) motivates a special case of this genre. We have accommodated all such constrained formulations in a software package designed for transportation planning purposes as reported in §5.

4. Illustrative Example

Consider the example presented by Li et al. (2001), pertaining to the classification tree T that is depicted in Figure 3. The number displayed in parentheses within each node is the node index, while the number below it is an error measure. Suppose that we restrict the number of leaf nodes to five, while minimizing the total resultant error measure. Noting (2), this would lead to the following optimization model:

$$\begin{aligned}
\text{Minimize} \quad & F_1(z) \equiv 38z_0 + 16z_1 + 13z_2 + 10z_3 \\
& \quad + 9z_5 + 3z_6 + 2z_7 + 5z_8 + 4z_9 \\
& \quad + 3z_{10} + 2z_{11} + 2z_{12} && \text{(18a)} \\
\text{subject to} \quad & z_7 + [z_3 + z_1 + z_0] = 1, && \text{(18b)} \\
& z_{11} + [z_8 + z_3 + z_1 + z_0] = 1, && \text{(18c)} \\
& z_{12} + [z_8 + z_3 + z_1 + z_0] = 1, && \text{(18d)} \\
& z_4 + [z_1 + z_0] = 1, && \text{(18e)}
\end{aligned}$$

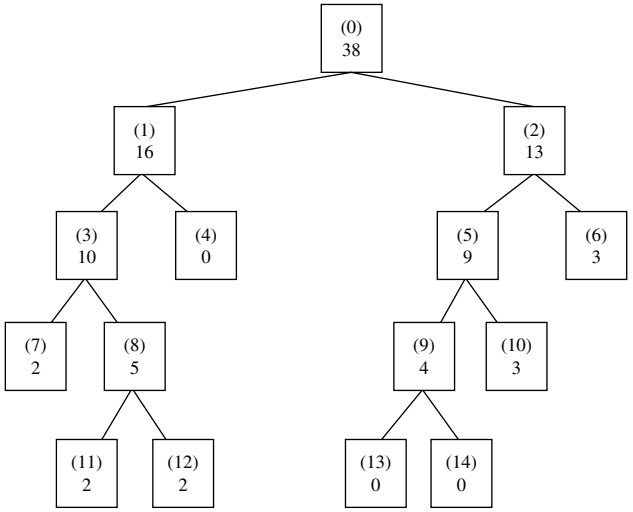


Figure 3 **Classification Tree for the Illustrative Example**

$$z_{13} + [z_9 + z_5 + z_2 + z_0] = 1, \tag{18f}$$

$$z_{14} + [z_9 + z_5 + z_2 + z_0] = 1, \tag{18g}$$

$$z_{10} + [z_5 + z_2 + z_0] = 1, \tag{18h}$$

$$z_6 + [z_2 + z_0] = 1, \tag{18i}$$

$$\sum_{h=0}^{14} z_h = 5, \tag{18j}$$

$$z \text{ binary}, \tag{18k}$$

where the objective function coefficients in (18a) are the error measures, constraints (18b)–(18i), and (18k) represent system (2), and constraint (18j) restricts the total number of leaf nodes to equal five. This problem has two alternative optimal solutions given, respectively, by z^{1*} and z^{2*} identified below, having an objective value of 19:

$$z_4^{1*} = z_5^{1*} = z_6^{1*} = z_7^{1*} = z_8^{1*} = 1, \tag{19a}$$

$$z_2^{2*} = z_4^{2*} = z_7^{2*} = z_{11}^{2*} = z_{12}^{2*} = 1. \tag{19b}$$

Note that this agrees with the solution given by Li et al. (2001) for the stage corresponding to five leaf nodes, but is obtained directly, without necessarily having to simultaneously discover the pruned trees for all other possible values of the number of leaf nodes. Furthermore, note that the CCP method would skip the case of five leaf nodes as discussed by Li et al. (2001) and would therefore not offer any solution to this particular problem.

Next, consider a secondary objective function $F_2(z) \equiv \sum_{h=0}^{14} c_h z_h$, with $c_h = 1\ \forall h \in \widetilde{T}$, and $c_h = 0$ otherwise, and suppose that we wish to minimize $F_1(z)$ given by (18a) with the first priority, and among all alternative optimal solutions, we would like to minimize $F_2(z)$. We can solve this preemptive priority problem via a composite objective function of the type to

$$\text{Minimize}\{F_1(z) + \lambda F_2(z)\colon (18b-k)\}, \tag{20}$$

where from the analysis of Sherali and Soyster (1983), we can take $\lambda < 1/(F_{2\max} - F_{2\min})$, where $F_{2\min} \le F_2(z) \le F_{2\max}\ \forall z$. For example, taking $\lambda = 0.1 < 1/(8-0)$, and solving (20) using the software CPLEX (2005), we directly obtain the desired solution z^{1*} given by (19a). Alternatively, after having found the optimal value of 19 for the objective function F_1, we could next solve the following problem as discussed in §3.1: Minimize $\{F_2(z)\colon F_1(z) = 19, (18b-k)\}$, which again produces the solution given by (19a).

5. Transportation Planning Case Study and Computational Results

In §2, we have described an approach for formulating suitable 0–1 optimization models to conduct the pruning of decision trees. Various objective functions along with desired side-constraints can be used within this context as discussed in §3. In this section, we present some details and results for our proposed methodology as applied to a large-scale transportation planning system called TRANSIMS (*transportation analysis and simulation system*), which is an integrated package of travel forecasting models and is designed to provide transportation planners with comprehensive information on traffic impacts, congestion, and pollution (Los Alamos National Laboratory 2002a).

In particular, the Activity Generator module in TRANSIMS develops a list of activities for each member of a synthetic household over a 24-hour horizon. The training data D in this context are comprised of *survey households* for which information related to demographic and activity characteristics for each household member is available. These attributes, which are used to partition the internal nodes of the tree at the different levels, include the household size, income, density, the number of vehicles owned, and the number of household members in different specified age groups. The CART algorithm is applied within TRANSIMS to build a classification tree, by virtue of which each survey household is effectively placed into one of the tree's terminal nodes. Another synthetic data set D', comprising the so-called *synthetic households*, is then generated based on certain available census information pertaining to the metropolitan area under study. By assigning D' to the leaf nodes of the developed tree T according to the demographic branching decisions in T, each synthetic household is matched with a survey household,

and this matching is then used to generate activity patterns for the members of the synthetic households over a 24-hour horizon.

For this case study using TRANSIMS, we investigated the use of different objective functions and side-constraints as delineated in §3 to optimally prune the generated classification tree. Our implementation uses JAVA along with the MySQL server version 4.0 (http://www.mysql.com/) and lp solve 2.0 (Berkelaar 1997), which is written in JAVA under the Lesser GNU Public License. The developed software, called CART-VT, allows a user to load the survey data in an Excel file format. Using the survey (training) data set comprised of demographic information for 3,470 households that was available from studies conducted on the Portland, Oregon transportation network (see Los Alamos National Laboratory 2002b), we constructed the tree T using the CART algorithm. This was accomplished in about 34 minutes for our survey data on a Dell 400SC 2.8 GHz computer having 1 GB of memory, using Rehat Linux 7.3, and it produced a tree T having 100 leaves and 199 nodes. Subsequently, for illustrative purposes, we matched some 23,656 synthetic households (comprising the synthetic data set D') to the survey households residing at the leaf nodes, while optimally pruning this tree using different practical compositions of objective functions and side-constraints of the type discussed in §3. The computational effort for all these runs (a sample of which is described below) remained under five CPU minutes, including the effort for data manipulation and model generation. (Note that a major component of this effort is due to manipulating the data for the given 23,656 synthetic households comprising D'. As evidenced by the results reported in §2, the proposed optimization model itself can be solved very quickly using an efficient solver such as CPLEX 2005.)

For the pruning step in our software, CART-VT, the user can select to minimize the total error function (using f_1 in (5)); minimize the total number of leaf nodes (using f_2 in (6a)); minimize the maximum depth of the leaf nodes (using f_3 in (6c)); maximize the minimum sum of values for some selected characteristic variable (designated as Y_{j^*} in §3.1) over the leaf nodes (using f_4 in (8) along with Problem PP3 in (9); here, for example, Y_{j^*} might represent the total number of trips for any household or data point); minimize the sum over all the leaf nodes of the ratios of the total number of synthetic households and the survey households that fall into the same leaf node (using f_5 in (10)); minimize the maximum over the leaf nodes of the ratio of the total number of synthetic households and the survey households that fall into the same leaf node (using Problem PP4 in (11), for example); or optimize any weighted sum of the available objective functions, where the total weight sums to one.

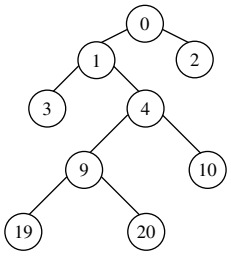


Figure 4 The Optimally Pruned Tree for Illustration 1

The side-constraints that are accommodated within CART-VT impose a bound on the total number of leaf nodes (via (16)); a bound on the depth of each leaf node (via (6b) or (15)); a bound on the ratio of the error (with respect to the survey data) of the new tree and that of the original tree (see (12)); and a bound on the ratio of the total number of synthetic households and the survey households that fall into the same leaf node (for each leaf node) (see (16)). The user can select a lower bound, an upper bound, or both for each side-constraint. In addition, the user can select to either preserve or prohibit some special original leaf nodes to be leaf nodes in the optimally pruned tree (see (17)).

We investigated different models by composing suitable weighted objective functions with various appropriate combinations of side-constraints. Below, we provide two illustrations.

Illustration 1. Suppose that we minimize the total number of leaf nodes while preserving a certain original leaf node that contains some special demographic characteristic, for example, householders comprised of only two workers having a total income of no more than \$50,000, and each being of age less than 30 years. This special demographic characteristic falls into node 19 (of the 199 nodes in T). The optimally pruned tree is shown in Figure 4. This same pruned tree is obtained when the objective is to minimize the maximum depth of the leaf nodes while preserving node 19.

Illustration 2. Another objective function that might be of interest from a statistical viewpoint is to minimize the error measure (with respect to the survey data) of the new tree, subject to the side-constraint that the ratio of the total number of synthetic households and the survey households that fall into the same leaf node be less than or equal to some value (say, seven) for all the leaf nodes. The resulting tree (not depicted because of its complexity) provides a matching that does not duplicate the generated activity patterns more than seven times. If we impose an additional side-constraint that limits the depth of each leaf node to be less than or equal to five to limit

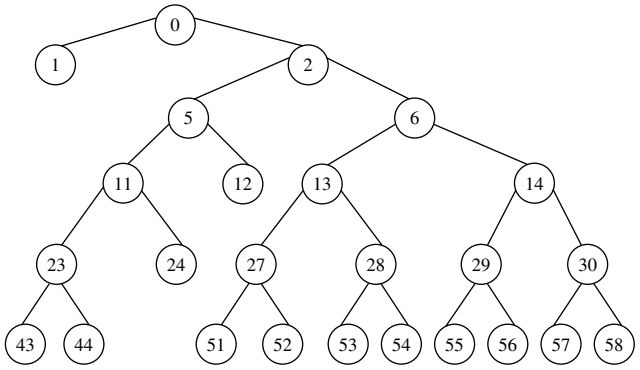


Figure 5 The Optimally Pruned Tree for Illustration 2

the extent to which the nodes are sequentially partitioned, the optimally pruned tree obtained is shown in Figure 5.

6. Summary and Conclusions

We have presented a 0–1 programming approach to optimally prune decision trees, while considering a variety of (possibly multiple) objective functions, as well as side-constraints that control the size, statistical characteristic, and structural properties of the resultant tree. We have shown that the basic model that seeks to minimize some general penalty function possesses a totally unimodular interval matrix-based set-partitioning structure, which is transformable to a shortest-path problem on an acyclic graph. Moreover, this model with an additional constraint governing the number of leaf nodes in the final tree can be solved in $O(mn)$ time, where m is the number of leaf nodes in the original tree. For more generally constrained problems, we have exhibited empirically that the underlying model structure facilitates the efficient solution of large-scale problems using an off-the-shelf commercial software such as CPLEX (2005). To motivate this modeling approach, we have also described a variety of objective functions and side-constraints that can be used in practice, formulated in terms of the model variables that are predicated on the important key feature of whether or not a node turns out to be a leaf node. This has been illustrated through a numerical example, as well as in the context of a real-life transportation planning system.

For future research, we recommend the testing of this optimal constrained pruning strategy in different data mining and classification contexts where decision trees are utilized in practice, including the formulation of specialized merit/penalty objective functions and side-constraints that are appropriate for such particular applications. Another avenue of useful research is to explore modeling frameworks for jointly performing the tree-growing and pruning strategies subject to various side-constraints in composing an optimal decision tree (see Rastogi and Shim 1998 for some guidelines along this direction).

Acknowledgments

This research was supported by the National Science Foundation under Grant DMI-0094462 and by the Federal Highway Administration. The authors gratefully acknowledge the constructive comments of the associate editor and a referee that have led to a significant improvement in the contents of this paper, as well as the assistance of Ahmed Ghoniem with some of the computations.

Appendix. Proof of Proposition 1

Let $T_{\text{Prune}}(E)$ be a pruned tree of T having leaf nodes E, and define the binary vector $z \in \{0,1\}^{n+1}$ to have components $z_h = 1\ \forall h \in E$, and $z_h = 0$ otherwise. Let us show that z satisfies (2a). Toward this end, consider any $h \in \widetilde{T}$ and let us examine the chain $C[h \to 0]$ in T. Let v be the first node in this chain that exists within $T_{\text{Prune}}(E)$ (see Figure A.1(a)). If $v = h$ itself, then $v \equiv h \in E$, while $E \cap p(h) = \varnothing$, and so, $z_h = 1$ and $z_u = 0\ \forall u \in p(h)$. Hence, (2a) holds true. Otherwise, if $v \neq h$, then we know that the nodes on $C[h \to v)$, in particular, are all collapsed nodes. Let $v' \in s_I(v)$ be the immediate successor of node v that lies on $C[h \to v]$ (possibly, $v' \equiv h$). Because v' is a collapsed node, we have that $b(v')$ is also a collapsed node by condition (ii) of Definition 1. In fact, together with condition (i) of Definition 1, this implies then that $v \in E$, while $E \cap p(v) = \varnothing$. Consequently, $z_v = 1$ while $z_h = 0$ and $z_u = 0\ \forall u \in p(h),\ u \neq v$. Hence, again we have that (2a) is satisfied.

Conversely, let z be any feasible solution to (2) and define E and τ as in (3a) and (3b). Hence, by construction, τ is a subtree of T that contains the root node 0. Next, let us show that E is precisely the set of leaf nodes of τ. Again, by the construction process (3b), we know that the set {leaf nodes of τ} $\subseteq E$. On the contrary, if any $e_1 \in E$ is not a leaf node of τ, then there must exist some $e_2 \in E$ for which $e_1 \in C(e_2 \to 0]$. Moreover, by tracing along successor nodes in T starting from the node e_2, let $e_3 \in \widetilde{T}$ be such that $e_2 \in C[e_3 \to 0]$ (possibly, $e_3 \equiv e_2$). By applying (2) to e_3, because $\{e_1, e_2\} \subseteq \{e_3\} \cup p(e_3)$, we must have that $z_{e_1} + z_{e_2} \leq 1$, which is a contradiction, because by assumption, $\{e_1, e_2\} \subseteq E$ implies that $z_{e_1} = z_{e_2} = 1$. Hence, $E \equiv$ {leaf nodes of τ}, and so, τ satisfies condition (i) of Definition 1.

To complete the proof, let us verify that condition (ii) of Definition 1 holds true by showing that for any node $h \in \tau$, $h \neq 0$, we also have that $b(h) \in \tau$. Toward this end, consider any $h \in \tau$, $h \neq 0$ and let v be the immediate predecessor of h on $c(h \to 0]$ (see Figure A.1(b)). Because $C[v \to 0] \cap E = \varnothing$, we have from (3a) that

$$z_u = 0 \quad \forall u \in C[v \to 0]. \tag{21}$$

Now, by tracing successor nodes in T starting from $b(h)$, suppose that we reach a leaf node $e \in \widetilde{T}$ (see Figure A.1(b)), so that

$$C[e \to 0] \supseteq C[b(h) \to 0] \equiv b(h) \cup C[v \to 0].$$

From (2), we have

$$z_e + \sum_{u \in C(e \to b(h))} z_u + z_{b(h)} + \sum_{u \in C[v \to 0]} z_u = 1. \tag{22}$$

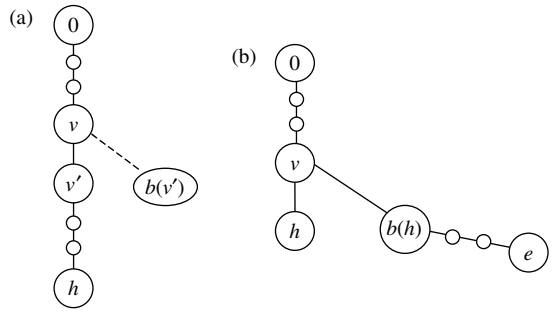


Figure A.1 Illustration for the Proof of Proposition 1

Therefore, on the contrary, if $b(h) \notin \tau$, then because we have shown above that E is the set of leaf nodes of τ, we would have that $b(h) \notin E$, and so by (3a), $z_{b(h)} = 0$. Moreover, we similarly have that $C[e \to b(h)] \cap \tau = \varnothing$, and so,

$$z_e + \sum_{u \in C(e \to b(h))} z_u + z_{b(h)} = 0.$$

This, together with (21), asserts that the entire left-hand side in (22) is zero, a contradiction. Hence, $b(h) \in \tau$, and this completes the proof. □

References

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ.

Bazaraa, M. S., J. J. Jarvis, H. D. Sherali. 2005. *Linear Programming and Network Flows*, 3rd ed. John Wiley & Sons, New York.

Bennett, K. P. 1992. Decision tree construction via linear programming. *Proc. 4th Midwest Artificial Intelligence Cognitive Sci. Soc. Conf., Utica, IL,* Southern Illinois University Carbondale, Carbondale, 97–101.

Bennett, K. P. 1994. Global tree optimization: A non-greedy decision tree algorithm. *Comput. Sci. Statist.* **26** 156–160.

Berkelaar, M. 1997. lp_solve 2.0. http://www.cs.wustl.edu/~javagrp/help/LinearProgramming.html.

Bertsekas, D. P. 2001. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA.

Bredensteiner, E. J., K. P. Bennett. 1998. Feature minimization within decision trees. *Comput. Optim. Appl.* **10**(2) 111–126.

Breiman, L., J. H. Friedman, L. A. Olshen, C. J. Stone. 1984. *Classification and Regression Trees*. Wadsworth & Brooks, Pacific Grove, CA.

Breslow, L. A., D. W. Aha. 1997. Simplifying decision trees: A survey. *Knowledge Engrg. Rev.* **12**(1) 1–40.

Chou, P. A., T. Lookabaugh, R. M. Gray. 1989. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. Inform. Theory* **35**(2) 229–315.

CPLEX. 2005. Using the CPLEX Linear Optimizer, version 9.0. CPLEX Optimization, Inc., Incline Village, NV.

Folino, G., C. Pizzuti, G. Spezzano. 1999. A cellular genetic programming approach to classification. *Proc. Genetic and Evolutionary Comput. Conf., GECCO99, Orlando, FL*, Morgan Kaufmann, San Francisco, 1015–1020.

Folino, G., C. Pizzuti, G. Spezzano. 2001. Parallel genetic programming for decision tree induction. *Proc. 13th Internat. Conf. Tools with Artificial Intelligence, Dallas*, IEEE Computer Society, Washington, D.C., 129–135.

Folino, G., C. Pizzuti, G. Spezzano. 2002. Improving induction decision trees with parallel genetic programming. *Proc. 10th Euromicro Workshop on Parallel, Distributed Network-based Processing*, IEEE Computer Society, Washington, D.C., 181–187.

Fu, Z., B. Golden, S. Lele, S. Raghavan, E. Wasil. 2003a. A genetic algorithm-based approach for building accurate decision trees. *INFORMS J. Comput.* **15**(1) 3–22.

Fu, Z., B. Golden, S. Lele, S. Raghavan, E. Wasil. 2003b. Genetically engineered decision trees: Population diversity produces smarter trees. *Oper. Res.* **51**(6) 894–907.

Fu, Z., B. Golden, S. Lele, S. Raghavan, E. Wasil. 2006. Diversification for better classification trees. *Comput. Oper. Res.* **33**(11) 3185–3202.

Li, X., J. Sweigart, J. Teng, J. Donohue, L. Thombs. 2001. A dynamic programming based pruning method for decision trees. *INFORMS J. Comput.* **13**(4) 332–344.

Lin, J., J. A. Storer, M. Cohn. 1992. Optimal pruning for tree-structured vector quantization. *Inform. Processing Management* **28** 723–733.

Los Alamos National Laboratory. 2002a. Transportation Analysis Simulation System (TRANSIMS) version: TRANSIMS-LANL-3.0. Los Alamos, NM.

Los Alamos National Laboratory. 2002b. Transportation Analysis Simulation System (TRANSIMS): Portland Study Reports. Los Alamos, NM.

Mehta, M., J. Rissanen, R. Agrawal. 1995. MDL-based decision tree pruning. *Proc. First Internat. Conf. Knowledge Discovery and Data Mining* (*KDD*), *Montreal,* AAAI Press, Menlo Park, CA, 216–221.

Murthy, S. K. 1997. On growing better decision trees from data. Ph.D. dissertation, University of Maryland, College Park.

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

Nemhauser, G. L., L. A. Wolsey. 1999. *Integer and Combinatorial Optimization*, 2nd ed. John Wiley & Sons, New York.

Quinlan, J. R. 1988. Simplifying decision trees. B. Gaines, J. Boose, eds. *Knowledge Acquisition for Knowledge-Based Systems*. Academic Press, London, 239–252.

Quinlan, J. R., R. L. Rivest. 1989. Inferring decision trees using the minimum description length principle. *Inform. Comput.* **80** 227–248.

Rastogi, R., K. Shim. 1998. PUBLIC: A decision tree classifier that integrates building and pruning. *Proc. 24th VLDB* (*Very Large Data Bases*) *Conf., New York,* Morgan Kaufman Publishers, San Francisco, 404–415.

Safavin, S. R., D. Landgrebe. 1991. A survey of decision tree classifier methodology. *IEEE Trans. Systems, Man, Cybernetics* **21**(3) 660–674.

Sherali, H. D. 1991. On the equivalence between some shortest path problems. *Oper. Res. Lett.* **10**(2) 61–65.

Sherali, H. D., A. L. Soyster. 1983. Preemptive and nonpreemptive multi-objective programs: Relationships and counter examples. *J. Optim. Theory Appl.* **39**(2) 173–186.

Ye, N. 2003. *Handbook of Data Mining*. Lawrence Erlbaum Associates, Mahwah, NJ.

Zhang, Y., H. Huei-chuen. 2005. Decision tree pruning via integer programming. Working paper, Department of Management Sciences, University of Iowa, Iowa City.