M2AN - MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE

R. S. BUCY
R. S. DIESPOSTI

Decision tree design by simulated annealing

M2AN - Modélisation mathématique et analyse numérique, tome 27, n° 5 (1993), p. 515-534

<http://www.numdam.org/item?id=M2AN_1993__27_5_515_0>

MATHEMATICAL MODELLING AND NUMERICAL ANALYSIS
MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE

(Vol. 27, n° 5, 1993, p. 515 à 534)

DECISION TREE DESIGN BY SIMULATED ANNEALING (*)

by R. S. BUCY (1) and R. S. DIESPOSTI (2)

Communicated by R. TEMAM

Abstract. — *In this research the simulated annealing algorithm is applied to design efficient classification and decision trees. Simulated annealing is a random search optimization algorithm. Other researchers have used the algorithm on similar types of combinatorial problems. For simple cost criteria, designs are obtained which match or improve upon those of an Information Theory greedy algorithm. Optimal solutions for several different cost functions are demonstrated along with cost efficient, robust designs that handle misclassification error.*

Résumé. — *Dans ce travail, l'algorithme du recuit simulé est appliqué à la recherche de classifications et d'arbres de décisions efficaces. La méthode du recuit simulé est un algorithme d'optimisation par recherche aléatoire. D'autres auteurs ont utilisé cet algorithme pour des problèmes combinatoires du même type. Pour des critères de coût simples, on obtient des algorithmes équivalents ou supérieurs à ceux qui reposent sur la théorie de l'information. On présente des solutions optimales pour différents choix de fonctions de coût, ainsi que des algorithmes robustes et efficaces pour traiter les erreurs de classification.*

1. INTRODUCTION

In a binary identification problem, a set of binary tests, or questions with *yes/no* answers, are applied to identify an object. A finite set of objects is described by specifying the test outcomes (yes or no) for each test for each object (†). The tree design problem then consists of the construction of an

(*) Manuscrit reçu le 10 juin 1992.

This work was supported by the Aerospace Corporation's Sponsored Research (ASR) Program.

(1) Professor of Aerospace Engineering and Mathematics, Department of Aerospace Engineering, University of Southern California, Los Angeles, California, USA 90089-1191. Fax : (213) 740-7774.

(2) Member of the Technical Staff, The Aerospace Corporation, Mail Stop M4/949, P.O. Box 92957, Los Angeles, California, USA 90009-2957. Fax : (310) 336-5827, e-mail : diespos@aerospace.aero.org.

(†) This information can be stored in tabular form as an *incidence matrix*.

M² AN Modélisation mathématique et Analyse numérique 0764-583X/93/05/515/20/$ 4.00

efficient testing procedure for identifying some unknown object belonging to the set.

Mathematically, the binary decision problem can be described as follows. A finite set of objects, $\mathbf{O} = \{O_1, O_2, \ldots, O_N\}$ is represented by the answers to a finite set of binary questions or tests, $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_M\}$. Each object belongs to a class, the set of classes being $\mathbf{C} = \{C_1, C_2, \ldots, C_K\}$, where $K \leqslant N$. In general, several objects may belong to each class. Using this information, construct an *efficient* testing procedure (or classification tree). Given an unknown object belonging to $\mathbf{O}$, apply the testing procedure to determine which class contains that object. Only the class information is the desired output from the procedure, the actual identity of the object is immaterial. This problem reduces to a binary object identification problem [9] when there is a one-to-one correspondence between the set of objects and the set of classes.

Results of this research can be applied in such areas as diagnostic systems design, species identification, the design of data processing algorithms, pattern recognition, and expert systems design [9], [15], [11], since these problems can often be represented by the previous mathematical model.

An *optimal* tree is one that minimizes some performance criterion. For example, the design criterion for a real-time expert system may be *average central processing unit (CPU) time* or *probability of an erroneous decision.* For a diagnostic system, the criterion may be *average testing time to identify the faulty part.* For the design of data processing algorithms, the criterion may be *storage required to code the testing procedure.* Some robust designs may seek to incorporate two or more of these criteria.

Though algorithms exist for the design of decision trees, no practical one guarantees optimality for problems of significant size. Simulated annealing is a numerical optimization algorithm [13]. In 1953 Metropolis *et al.* [14] defined and implemented an early version of this algorithm. In metallurgy, annealing is the process whereby a metal is first liquified then slowly cooled. As the metal cools, the atoms form a lattice, a minimum energy configuration.

The objective is to apply the simulated annealing algorithm to find a tree that minimizes a specified cost criterion for some specified decision problem. In the tree design problem, a tree may be considered a function of the question configuration, i.e. the questions specified at its nodes. The algorithm searches for a minimum by a random walk over the configuration space, in our case by randomly reassigning questions to the nodes of a tree. Simulated annealing has been successfully applied to such combinatorial problems as the traveling salesman problem [13], [1], communication code design [7], and computer circuit design [13].

Since the construction of optimal decision trees is an *NP-complete* problem [12], a more feasible objective for large problems would be to find

an efficient *near-optimal* tree. For some real-world problems, a near-optimal solution that satisfies other characteristics may even be preferable to an optimal solution. For example, for real-time application, a *fast* expert system may be highly desirable, but such a design would be of little practical value if it exhibits a large error rate.

2. SIMULATED ANNEALING APPLIED TO TREE DESIGN

Simulated annealing is a numerical optimization algorithm that simulates the physical process of annealing ([2]). The algorithm operates to find a state or configuration that minimizes some objective or cost function of the state. The temperature T is a control parameter in the algorithm, which is usually adjusted by some systematic *cooling schedule*. The starting temperature is high enough to allow the system to *melt*, i.e. T sufficiently high to allow jumps over the entire search space.

At each temperature, the algorithm iterates by taking random steps in the configuration space. Following each step, a decision is made whether to stay at the current state, or accept the new state. Let $\Delta C = C_{\text{new}} - C_{\text{current}}$ be the change in cost due to the random step. If $\Delta C \leq 0$, the new state is always accepted since it has lower cost. However, if $\Delta C > 0$, the new state is accepted with probability $B_A = \mathbf{e}^{-\frac{\Delta C}{T}}$, the Boltzmann acceptance criterion [13]. Operationally, a uniform random number is generated, $\zeta \in [0, 1]$. If $\zeta < B_A$, then the new state is accepted, otherwise the current state is kept. This strategy, which allows the acceptance of a higher cost configuration, enables the process to escape local minima.

Many random steps are taken at each temperature, and the temperature is lowered in an outer loop. At high temperatures, the quantity $\mathbf{e}^{-\frac{\Delta C}{T}}$ is usually close to one for $\Delta C > 0$. Thus a higher cost state is usually accepted at high temperatures. As the temperature lowers, the system freezes, and higher cost configurations are accepted with diminishing probability. Also as the temperature lowers, the algorithm converges to the optimum or a state whose cost is close to optimal.

Simulated annealing has provided solutions to some combinatorial optimization problems that are better than any previously found [7]. When tested on some ill-conditioned functions of continuous variables, the algorithm gave improved results ([3]) over the *Nelder and Mead simplex method* and an *Adaptive Random Search* algorithm [5].

([2]) The procedure to drive a physical system, such as steel, into a minimal energy configuration by first heating to melt, then slowly lowering the temperature.

([3]) Solutions closer to the minimum, in some cases with fewer function evaluations.

Our simulated annealing algorithm searches over the set of trees of a fixed length. A tree is specified by an assignment of questions to the tree nodes. For M questions, full length trees have M questions along each branch, and 2^M branches, each corresponding to a different combination of binary answers. The top node of the tree is associated with the set of all of the objects. The test Q at the top node splits this set into two disjoint sets : those objects which test *yes* for Q and those which test *no* for Q. These two subsets are then associated with two nodes at the second level. Questions acting at these two nodes then further subdivide each corresponding subset. This splitting operation is continued until single objects are associated with terminal nodes. Only paths which lead to terminal nodes are retained in the tree structure. Other branches are deleted. We call this process the *pruning* of a complete M question tree.

To decrease time and memory requirements, we can also look at reduced length trees whose branches have length $M_R < M$. For a given fixed length tree sometimes an assignment of objects to terminal nodes cannot be found which satisfies the incidence matrix. This can occur when (1) one object is defined more than once by different combinations of *yes/no* answers, (2) the same combination of answers defines two or more objects from different classes, or (3) more than M_R questions may be needed to separate two objects from different classes. We assume that the incidence matrix is defined such that the first two conditions do not occur. However, the last case can occur for reduced length trees with length $M_R < M$. During processing, when this happens, the tree is labeled inconsistent since it does not represent a solution, and it is discarded.

For the decision problem, in which several objects can be associated with each class, an additional step is required to prune the tree. Recall from the Introduction that only the class which contains the object is the desired output of the procedure. Objects of the same class which are associated with *adjacent* terminal nodes can be merged ([4]). This merging operation is continued until all adjacent terminal nodes are associated with objects from distinct classes.

In our case, the objective function to minimize is some specified, arbitrary cost function on trees. The cost of a given fixed length tree can be calculated after it is pruned as described in the preceding paragraphs. The system state or tree configuration is set by specifying the questions acting at the nodes of a tree. Random reconfigurations are obtained by switching two questions chosen at random, or by reconstructing the entire tree or some part of it. In an outer loop, a new tree is obtained by a random reassignment of questions to all nodes or by keeping the same sequence of questions on one branch of the old tree and randomly reassigning the remaining questions. In an inner loop,

([4]) This is the inverse operation to splitting.

two questions, Q_i and Q_j, chosen at random are switched, i.e. if $Q_i(Q_j)$ is assigned to some node of an old tree, that node of the new tree will have question $Q_j(Q_i)$. Using these two methods of random reconfigurations, any tree in the complete set having branches of length M questions can be reached. Question assignment to tree nodes is constrained such that : (1) no questions are used more than once on a tree branch ([5]), (2) a question applied to some node must be chosen to split the object subset at that node.

A tree configuration is *pruned* by assigning classes to terminal nodes in agreement with the incidence matrix. A tree constructed in this manner, and which satisfies any additionally imposed constraints, represents a valid solution to the classification problem. The decision whether to accept a new solution is made, using the Boltzmann acceptance criterion, once its cost is calculated.

Many random tree reconfigurations are performed at each temperature and the temperature is lowered in an outer loop by the relation $T_{i+1} = \alpha T_i$ where $0.7 \leqslant \alpha \leqslant 0.99$. The number of iterations at each temperature required to obtain adequate solutions depends on the problem size. Asymptotic convergence to the set of optimal solutions is guaranteed if an infinite number of random steps are taken [1].

3. COST CRITERIA

The cost of a decision tree can be determined after the tree is pruned. It depends on the prior probabilities of occurrence of the objects. For example, the *average number of questions* cost, is given by

$$\bar{Q} = \sum_{i=1}^{N_O} \mathbf{PR}_i \, (\mathbf{IL}_i)$$

where

$\mathbf{N_0}$ is the number of objects

$\mathbf{PR}_i$ is the prior probability of object O_i and

$\mathbf{IL}_i$ is the number of questions on the branch of the pruned tree leading to object O_i. (For the decision problem, this is the number of questions required to identify the class which contains object O_i.)

Another cost criterion of interest is the *average testing cost*. For each question Q_j, let T_j represent the cost due to applying that test or asking that question. The average testing cost is given by

$$\bar{T} = \sum_{i=1}^{N_O} \mathbf{PR}_i \sum_{j \in B_i} T_j$$

([5]) This precludes the possibility of asking the same question more than once in some logical path.

where the inner sum is over all questions Q_j on the tree branch assigned to object O_i. If T_j is the time required to apply test Q_j, then this cost function is the average time to classify some unknown object. Also the *average testing cost* is more general than the *average number of questions* since $\bar{T} = \bar{Q}$ if $T_j = 1, \quad \forall j$.

For the object identification problem, the *probability of misclassification error* can be calculated by applying Bayes' Theorem

$$P_e = \sum_{i=1}^{N_O} P_{e|i} \, \mathbf{PR}_i$$

where

$P_{e|i}$ is the probability of error given object O_i.

For each object O_i, assuming independent errors,

$$P_{\text{correct}|i} = \prod_{k \in B_i} (1 - Pr(k|i))$$

where

the product is over all questions Q_k on the tree branch B_i assigned to object O_i

$Pr(k|i)$ is the probability that the answer to question Q_k is in error given object O_i (these values are specified as a matrix of input data).

Then

$$P_{e|i} = 1 - P_{\text{correct}|i} \, .$$

Designs which are robust with respect to misclassification error can be realized by adding the error probability to one of the other cost criteria.

4. KNOWN TREE DESIGN PROCEDURES AND BOUNDS

For M questions, the number of distinct binary trees of length M (trees with a unique assignment of questions) is [15]

$$N_T = \prod_{i=0}^{M-1} (M - i)^{2^i} \, .$$

In the next section, designs are obtained for seven questions ($N_T = 1.9 \times 10^{27}$) and for 25 questions ($N_T = 10^{7.4 \times 10^6}$). One approach to finding an optimal tree is to evaluate the cost for each tree in the solution space of size N_T and selecting a tree with minimal cost. However, considering the size of the set, this is not practical even for small problems.

Several greedy algorithms based on the entropy function of Information Theory have been developed [15], [11], [3]. Versions are usually cost

function specific. These algorithms design a tree from the *top-down*. Starting from the top node, at each node a question is assigned which optimizes some measure of information. The algorithm iterates down the nodes of the tree, assigning questions to further split the subset of objects until a terminal node is reached. Terminal nodes are associated with subsets of objects belonging to the same class. For the object identification problem (the case when the classes are the objects themselves) and the *average number of questions* cost, a very simple greedy algorithm exists [15]. At each node choose that question which splits the remaining set of objects into two subsets with most nearly equal probabilities. Though algorithms of this class are suboptimal [10], they are *fast* and often produce near-optimal or optimal designs for simple cost functions. A recent version seems to be applicable to arbitrary cost functions [16]. We speculate that their ability to incorporate additional constraints, and still provide near-optimal solutions, is limited.

Algorithms which guarantee an optimal design, such as the Branch and Bound algorithm [17] and dynamic programming [2] are not practical for large problems since execution times and/or storage grow exponentially with problem size. An elegant approach, which applies to cost criteria with a preorder relation, is able to find all minimal trees by applying syntactic optimization to generate irreducible terms [12].

The Huffman algorithm [9] is normally used to design optimal variable-length codes for messages with unequal probabilities [8]. The algorithm generates a code tree which assigns a binary code word to each message. Since objects of the incidence matrix are already assigned to terminal nodes, the Huffman tree is already pruned. However, an assignment of questions to the nodes of the tree, which splits the objects in agreement with the incidence matrix, may not exist. In contrast to the *top-down* approach of the greedy algorithms, the Huffman algorithm constructs a tree from the *bottom-up,* which is a characteristic of dynamic programming. Since the Huffman algorithm generates a minimal *average number of questions* tree, it provides an absolute lower bound which can be used to gauge the performance of other rules and design approaches. This bound is guaranteed to be attainable, however, only if a Huffman tree exists which is consistent with the incidence matrix.

5. RESULTS

Simulated annealing is applied to design trees for three illustrative problems. Additional examples and details are given in [4].

5.1. Seven-Segment Digit Recognition Problem

The seven-segment digit display consists of seven diodes arranged in the pattern shown in figure 1.

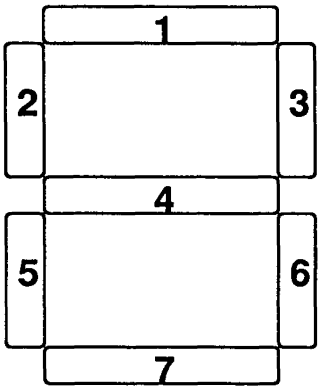

Figure 1. — Diode arrangement for the seven-segment digit recognition problem.

By turning on selected diodes, the digits **0-9** can be represented. This display is typical of that used in a pocket calculator. The objective is to identify some unknown digit from knowledge of the diode states, each being either *on* or *off*. The classes are the ten digits $\{0, 1, 2, \ldots, 9\}$ and the tests are the seven questions « *Is diode number j lit ?*, » $j = 1, 2, \ldots, 7$. The incidence matrix which specifies the states of the diodes for each digit follows [3].

Objects	Questions						
	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
$O_1 = 0$	1	1	1	0	1	1	1
$O_2 = 1$	0	0	1	0	0	1	0
$O_3 = 2$	1	0	1	1	1	0	1
$O_4 = 3$	1	0	1	1	0	1	1
$O_5 = 4$	0	1	1	1	0	1	0
$O_6 = 5$	1	1	0	1	0	1	1
$O_7 = 6$	1	1	0	1	1	1	1
$O_8 = 7$	1	0	1	0	0	1	0
$O_9 = 8$	1	1	1	1	1	1	1
$O_{10} = 9$	1	1	1	1	0	1	1

As an example, the digit 1 is represented by turning on diodes 3 and 6 while turning off the remaining diodes. Hence the row corresponding to digit 1 has a **1** (on) in the 3rd and 6th columns and a **0** (off) in the remaining columns.

Application of the Huffman algorithm gave a lower bound of 3.4 for the *average number of questions* cost criterion and equal object probabilities. An

Information Theory greedy algorithm produced a tree with cost of 3.4, which is guaranteed to be optimal by the Huffman bound.

Shown in figure 2 is an optimal tree as found by the simulated annealing algorithm. The initial temperature was 1.0, the final temperature was 0.01, and the geometric series factor was $\alpha = 0.80$. At each temperature, five tree reconfigurations, derived by switching two questions chosen at random, were performed in an inner loop, while five outer loop reconfigurations consisted of a random reassignment of all questions.

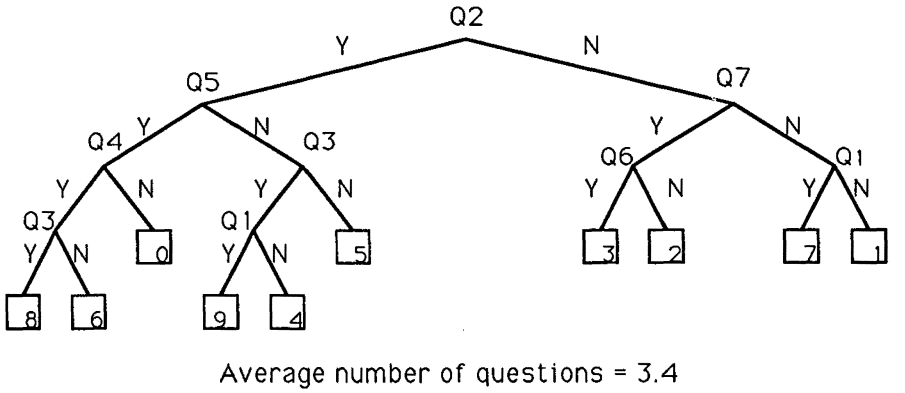

Figure 2. — Classification tree designed by simulated annealing for the seven-segment problem.

5.1.1. *Performance Analysis*

Presented following are Monte Carlo results to assess the performance of the simulated annealing algorithm as applied to the seven-segment problem. The numbers of outer loop reconfigurations (IOLL) and inner loop reconfigurations (IILL) at each temperature were varied parametrically. The initial temperature was 1.0, the final temperature was 0.01, and the cooling schedule factor was $\alpha = 0.75$. The statistics are based on 20 Monte Carlo runs. The percent error is $\varepsilon = \dfrac{\bar{Q} - \bar{Q}^*}{\bar{Q}^*} 100$ where the optimal cost is $\bar{Q}^* = 3.4$ for the equal probability case. Tabulated following are the average error and the average time per Monte Carlo run on a DEC 3100 workstation. Algorithm performance is seen to improve with the number of iterations. For eight iterations, all 20 of the Monte Carlo runs converged to optimal solutions.

IOLL = IILL	$\bar{\varepsilon}$ percent	average time per run sec
1	3.1	7.0
2	1.8	14.7
3	1.5	26.5
4	0.8	41.9
5	0.3	66.0
6	0.6	85.7
7	0.3	113.0
8	0.0	145.0

5.1.2. *Fault Tolerant Design*

To illustrate the effect of error, simulated annealing tree designs are obtained for a cost function that includes both the *average number of questions* cost and the *misclassification error*. The cost is given by

$$C = \sum_{i=1}^{N_O} \mathbf{PR}_i \, \mathbf{IL}_i + \beta P_e$$

where β is a relative weight.

Diode number 2 is assumed to be flickering and thus unreliable in the *on* state. To simulate this effect, the error probability matrix is assigned the values Pr (answer to Q_2 in error | object O_i) = 0.10, $i = 1, 5, 6, 7, 9, 10$, with all other elements equal to zero. Optimal tree designs are obtained as the weight β is varied parametrically. For small values of β, the optimal tree is a Huffman tree that applies question Q_2 first, and which has cost, as a function of β, of $C_H = 3.4 + 0.06\,\beta$. As β increases, the algorithm tends to converge to a tree that uses question Q_2 as few times as possible to classify the objects in the set $\{O_1, O_5, O_6, O_7, O_9, O_{10}\}$. Shown in figure 3 is an evolution of the optimal cost ([6]) as β varies. At $\beta = 2$ the optimal tree changes from a Huffman tree to a non-Huffman tree that uses question Q_2 only to classify object O_4 (no error contribution since $Q_2 = 0$ for object O_4) and object O_{10}. This tree structure remains optimal for $\beta > 2$ and its cost is given by $C_{NH} = 3.5 + 0.01\,\beta$. Since all six objects in the set $\{O_1, O_5, O_6, O_7, O_9, O_{10}\}$ contribute an error for the Huffman tree and since only object O_{10} contributes an error for the non-Huffman tree, the slope of the optimal cost *versus* β is six times greater for $\beta < 2$ than for $\beta > 2$. Shown in figure 3 are two optimal trees, one for each side of the transition weight.

([6]) The trees and the corresponding costs are believed to be optimal based on empirical results.

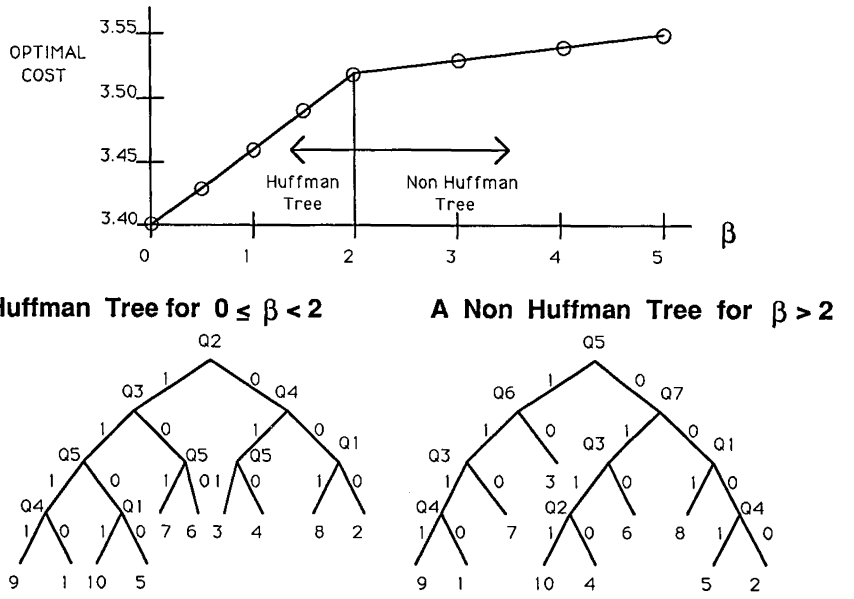

Figure 3. — Transition from Huffman to non-Huffman tree for flickering diode problem.

It is interesting to note that a similar type of example involving errors for the seven-segment problem is presented in [3]. In that case, each diode had a probability of 0.1 of having the erroneous state. Using this error probability, 200 object samples were generated as training data. A greedy algorithm applied to this random set produced a tree with 3.6 *average number of questions* and misclassification error rate of 0.30.

5.1.3 *Effect of Criterion Modification*

The following cost has application, for example, in the design of data processing algorithms which have a cycle time constraint. Suppose that in addition to *average number of questions,* the sample variance is also a design concern. The modified cost criterion is taken as

$$C = \bar{Q} + \beta \sigma_Q$$

where

$$\bar{Q} = \sum \mathbf{PR}_i \, \mathbf{IL}_i \text{ is the average number of questions}$$

$$\sigma_Q^2 = \sum \mathbf{PR}_i \, (\mathbf{IL}_i - \bar{Q})^2 \text{ is the sample variance}$$

of the number of questions .

The object prior probabilities are chosen as **PR** = [0.06, 0.06, 0.06, 0.06, 0.06, 0.26, 0.26, 0.06, 0.06, 0.06]. For $\beta = 0$ the cost reduces to *average number of questions* only, and an optimal tree (which is also a Huffman tree) as found by simulated annealing is shown in figure 4. For $\beta \in [0, \beta^*]$, this Huffman tree is also optimal ([6]) for the modified criterion, and its cost, as a function of β, is given by $C_H = \bar{Q} + \beta \sqrt{0.9984}$, where $\bar{Q} = 2.96$. However, when $\beta \geq \beta^*$ the optimal tree changes to a non-Huffman tree. One such tree is shown in figure 4, and its cost is given by $C_{NH} = \bar{Q} + \beta \sqrt{0.1824}$, where $\bar{Q} = 3.24$. We have tested values of β up to 10^5, and the optimal tree retains this same structure. The value of β at which the transition occurs is found by equating the two costs, $C_H = C_{NH} \Rightarrow \beta^* = \dfrac{0.28}{\sqrt{0.9984} - \sqrt{0.1824}}$. As β increases above β^*, it is clear that the cost of the non-Huffman tree becomes arbitrarily smaller than the Huffman tree cost.

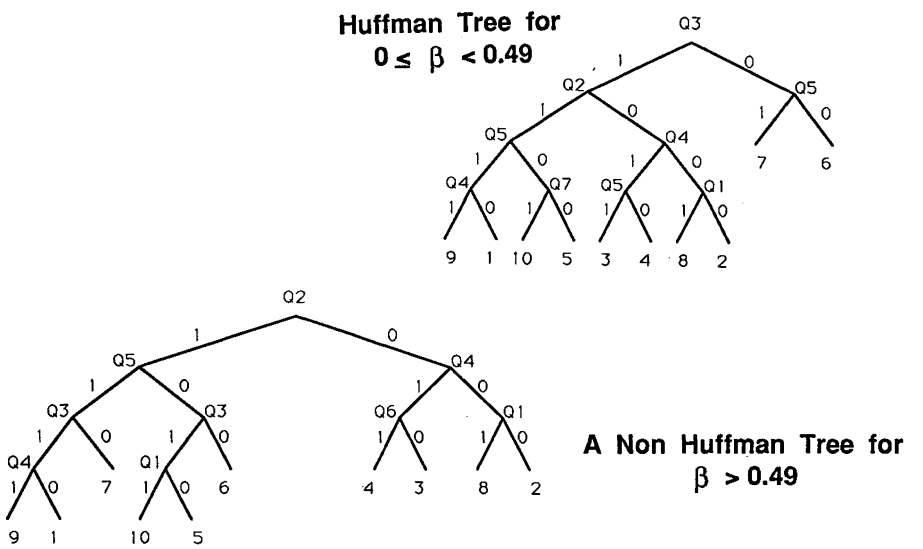

Figure 4. — Optimal trees for the seven-segment problem with average length plus standard deviation cost.

5.2. Average Testing Cost Problem

This problem is *Example 1* from reference [9]. The cost function is the average testing cost as defined in Section 3. The incidence matrix is given by

Objects	PR_i	Question costs 45 Q_1	150 Q_2	35 Q_3	60 Q_4	25 Q_5	180 Q_6	10 Q_7
O_1	0.05	0	0	0	1	1	0	0
O_2	0.10	0	0	0	0	1	0	0
O_3	0.10	1	0	1	0	0	0	0
O_4	0.05	1	1	1	0	0	1	0
O_5	0.10	1	0	0	1	1	0	0
O_6	0.10	1	0	0	0	1	0	0
O_7	0.40	0	0	0	1	0	1	1
O_8	0.10	1	0	1	1	0	0	0

Above each question Q_j is the cost T_j associated with asking that question. The optimal tree is derived in [9] and is shown in figure 5.

The simulated annealing algorithm, using 10 inner loop and 10 outer loop reconfigurations at each temperature, and a geometric series factor $\alpha = 0.80$, converged to this optimal tree.

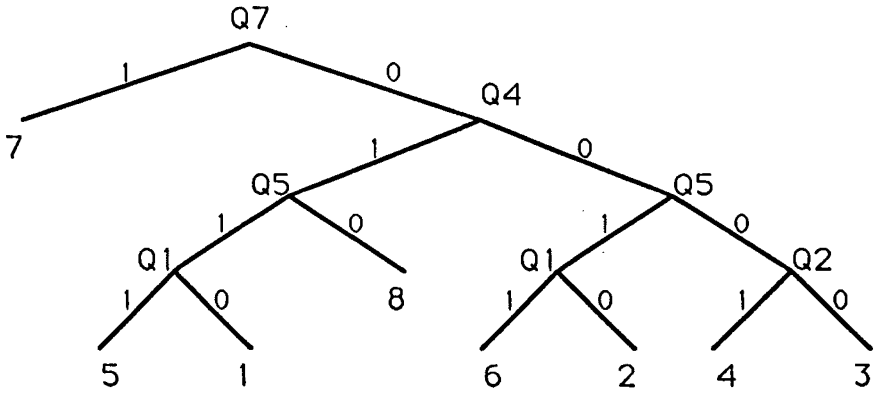

Figure 5. — Minimal average testing cost example.

5.3. The 5 x 5 Pixel Letter Identification Problem

A letter of the English alphabet is to be identified from the states of a grid of 5×5 pixels. There are $5 \times 5 = 25$ questions of the form « *Is pixel number j on* ? », $j = 1, 2, \ldots, 25$. Shown in figure 6 are the letter definitions in terms of the 25 pixels states. Also shown are the question numbers assigned to the pixels. The incidence matrix is derived as shown in figure 7, where the prior probabilities of the letters were determined based on the frequencies of occurrence [6].

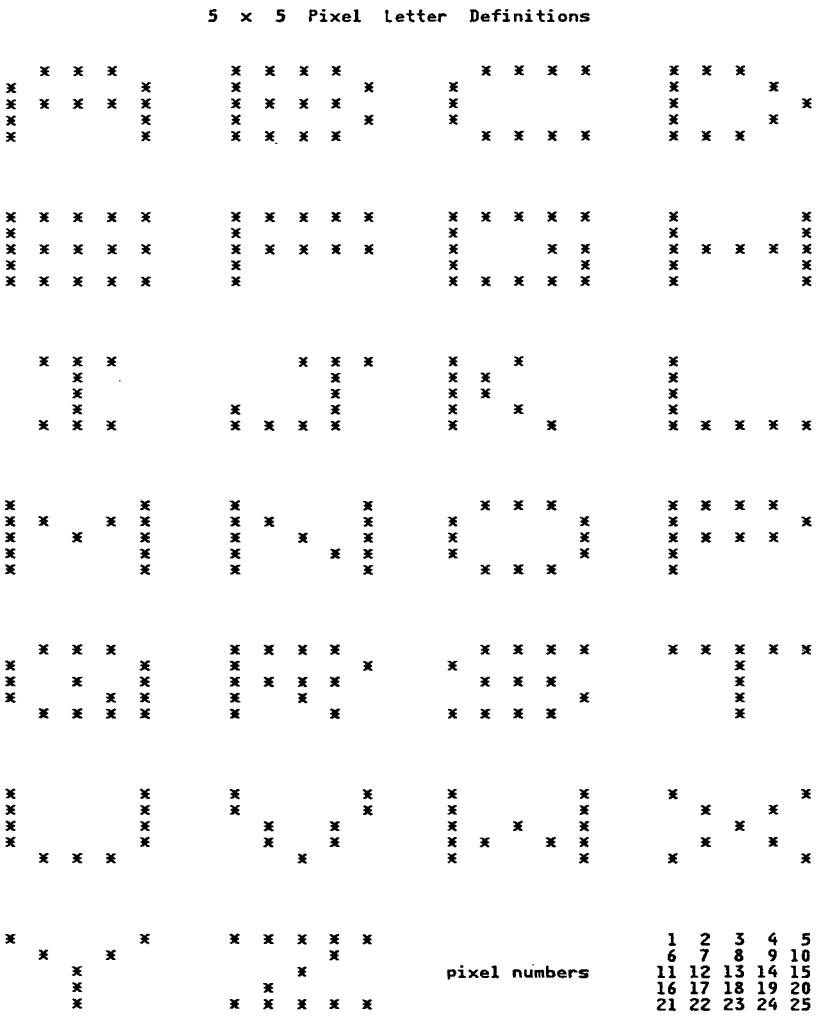

Figure 6. — 5 x 5 pixel letter definitions.

An absolute lower bound can be determined using the Huffman algorithm. As seen in figure 8, the lower bound for *average number of questions* is 4.1849. This bound is guaranteed to be tight, however, only if a Huffman tree consistent with the incidence matrix exists. For this problem, it can be shown that such a tree does not exist. Therefore this value is an absolute lower bound which may, however, be achievable by a non-Huffman tree.

Temperature histories of two simulated annealing runs are shown in figure 9 along with the costs of an Information Theory design (see next

Incidence Matrix for 5 x 5 Pixel Problem

Pr(j)	j	QUESTIONS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
.0788	1 - A	0	1	1	1	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0	1
.0156	2 - B	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0
.0268	3 - C	0	1	1	1	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	1	1
.0389	4 - D	1	1	1	0	0	1	0	0	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	0	0
.1268	5 - E	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1
.0256	6 - F	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0
.0187	7 - G	1	1	1	1	1	1	0	0	0	0	1	0	0	1	1	1	0	0	0	1	1	1	1	1	1
.0573	8 - H	1	0	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0	1
.0707	9 - I	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	1	0
.0010	10 - J	0	0	1	1	1	0	0	0	1	0	0	0	0	1	0	1	0	0	1	0	1	1	1	1	0
.0060	11 - K	1	0	1	0	0	1	1	0	0	0	1	1	0	0	0	1	0	1	0	0	1	0	0	1	0
.0394	12 - L	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1	1	1
.0244	13 - M	1	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	0	0	0	1	1	0	0	0	1
.0706	14 - N	1	0	0	0	1	1	1	0	0	1	1	0	1	0	1	1	0	0	1	1	1	0	0	0	1
.0776	15 - O	0	1	1	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0	1	1	1	0
.0186	16 - P	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1	0	0	0	0	1	0	0	0	0
.0009	17 - Q	0	1	1	1	0	1	0	0	0	1	1	0	1	0	1	1	0	0	1	1	0	1	1	1	1
.0594	18 - R	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1	0	1	0	0	1	0	0	1	0
.0631	19 - S	0	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1	1	0
.0978	20 - T	1	1	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
.0280	21 - U	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0	1	1	1	0
.0102	22 - V	1	0	0	0	1	1	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	1	0	0
.0214	23 - W	1	0	0	0	1	1	0	0	0	1	1	0	1	0	1	1	1	0	1	1	1	0	0	0	1
.0016	24 - X	1	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1
.0202	25 - Y	1	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
.0006	26 - Z	1	1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1	1

Figure 7. — Incidence matrix for the 5 x 5 pixel problem.

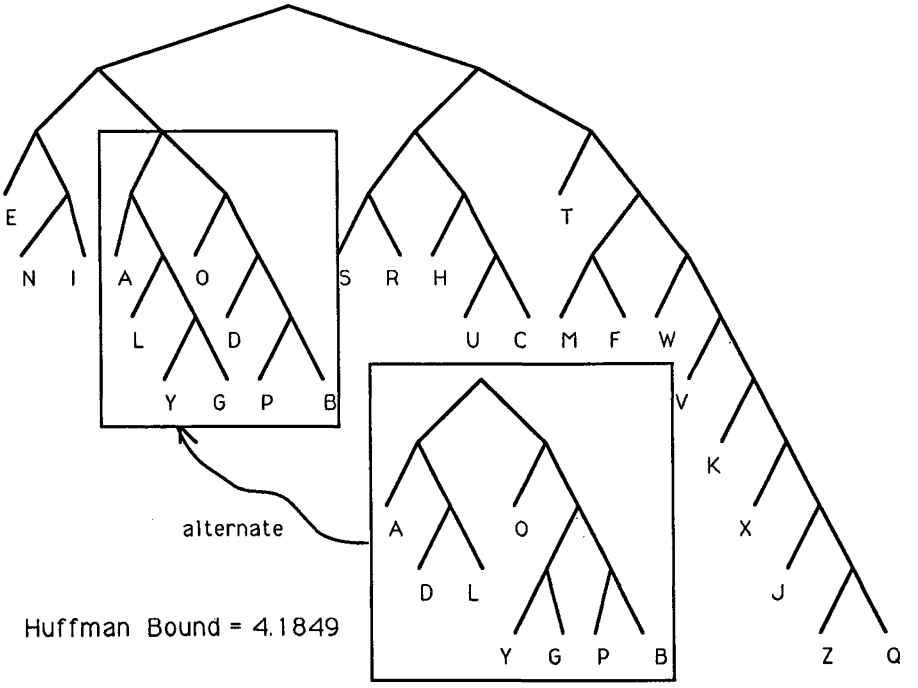

Figure 8. — Huffman tree for the letter identification problem.

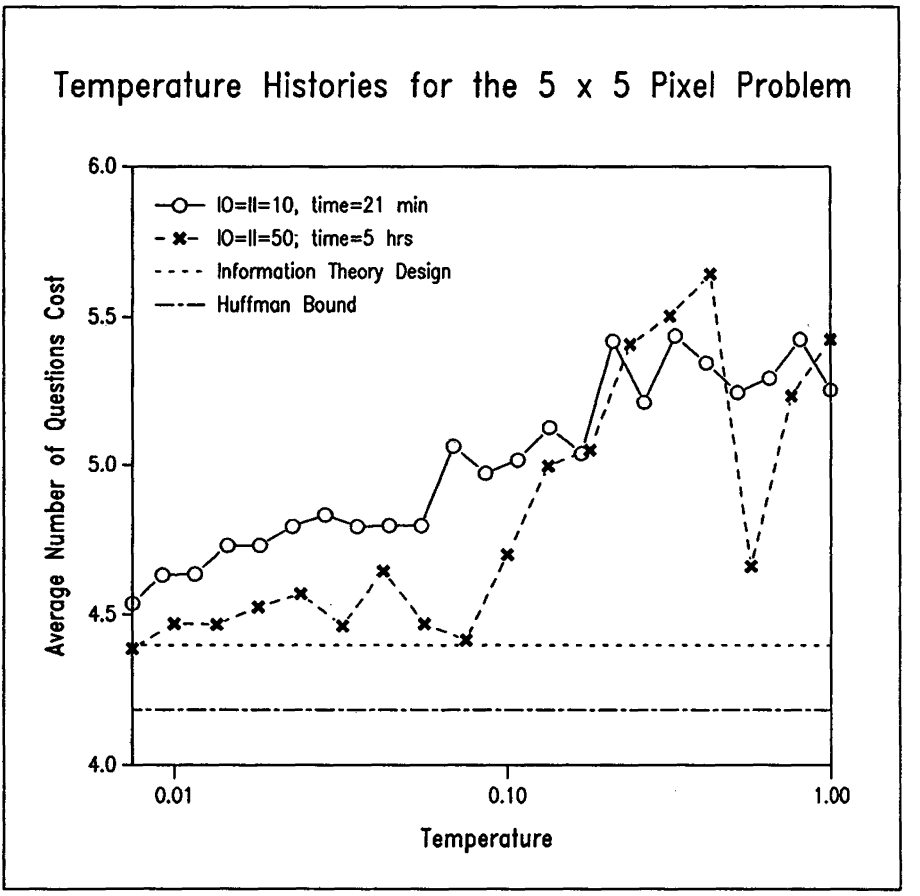

Figure 9. — Temperature histories for the 5 × 5 pixel problem.

paragraph) and the Huffman bound. Shown is the cost function evolution as the temperature decreases from 1.0 to 0.01, according to $T_{i+1} = \alpha T_i$. The first run utilized an α of 0.80 and ten outer loop and ten inner loop reconfigurations at each temperature. The second run, which considered trees of reduced length $M_R = 7$, used $\alpha = 0.75$ and 50 outer and inner loop reconfigurations, produced a tree with cost of 4.3879 Also shown in the legend are the processing times on a SUN SPARC station 1.

Shown in figure 10 is the tree as found by simulated annealing, having cost 4.3879. The size of the configuration space is $\approx 10^{165}$, which corresponds to the number of distinct trees of length seven questions, each question chosen from 25 possible questions. The ratio of the number of trees searched to the

total number of trees is $\frac{45,000}{10^{165}} = 5 \times 10^{-161}$. The Information Theory algorithm generated the tree shown in figure 11, with cost 4.3989.

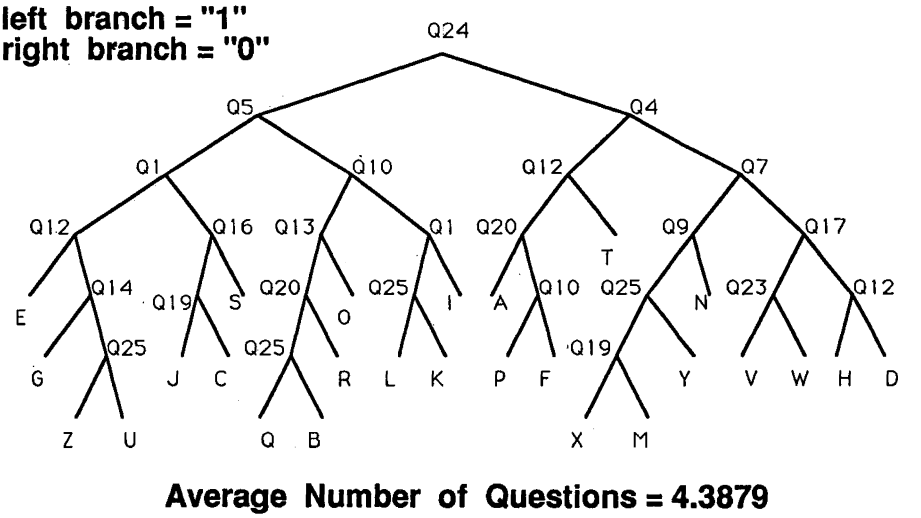

Figure 10. — 5 × 5 pixel tree designed by information theory.

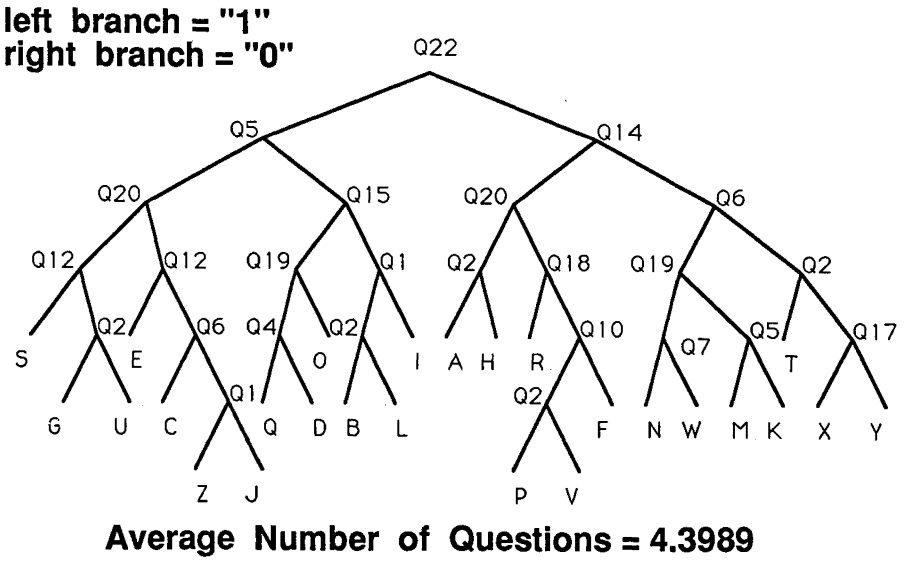

Figure 11. — 5 × 5 pixel tree designed by simulated annealing.

5.4. Summary of Results

The following table summarizes the performance and applicability of a greedy Information Theory algorithm, the Huffman algorithm, and simulated annealing, as applied to the example problems. The codes are : SubOpt - suboptimal design ; Opt - optimal design ; NA - the implemented algorithm does not apply directly ; Incon - the Huffman tree is inconsistent with the incidence matrix ; Opt ? - believed to be optimal based on empirical results ; Opt ? ? - best solution to date, but not known if optimal.

Summary of algorithm results			
	Greedy Algorithm	Huffman Algorithm	Simulated Annealing
Seven-Segment Problem	Opt	Opt	Opt
Seven-Segment with Errors	NA	NA	Opt ?
Seven-Segment with $\bar{Q} + \beta \sigma_Q$ Cost	NA	NA	Opt ?
Average Testing Cost Problem	NA	NA	Opt
5×5 Pixel Letter Problem	SubOpt	Incon	Opt ? ?

The greedy algorithm solution to the 5×5 letter problem is known to be suboptimal since simulated annealing found a lower cost tree.

6. CONCLUSIONS AND FUTURE EFFORT

Simulated annealing has been successfully applied to find optimal trees for the classical seven-segment digit recognition problem and an *average testing cost* problem. For the 5×5 pixel problem, simulated annealing produced a lower cost design than an Information Theory greedy algorithm.

The algorithm was also applied to find optimal designs for some non standard cost functions. In the fault tolerant design, a new cost criterion was formed by adding the weighted misclassification error probability to the

average number of questions. This design behaved as expected. As the weighting on the error probability became large, the algorithm converged to a tree that minimized the errors at the expense of a larger value of *average number of questions.* In the modified cost criterion design, the weighted standard deviation was added to the *average number of questions cost.* Both of these problems exhibited a transition of optimal tree structures at some value of the relative weight. We are attempting to characterize this transition phenomenon.

Recommendations for future effort are as follows. Methods to improve algorithm efficiency should be explored, such as strategies to reduce the search set of trees and methods to speed up the calculation of change in cost with respect to a randomly perturbed tree. Parallel implementations should be investigated. Application of the algorithm to expert systems design should be studied. It is anticipated that considerations inherent to decision systems, such as, constraints on question sequencing, costs due to *asking a question,* decision priorities, and penalities for erroneous decisions, can be easily integrated into the algorithm. Future research should study hybrid algorithms that merge the speed of the greedy algorithms with the adaptability of the simulated annealing approach.

The simulated annealing approach presents an effective way to perform a combinatorial search for optimal solutions and in contrast to *fast* heuristics based on Information Theory, has the versatility to accommodate constraints, arbitrary cost criteria, and other real-world types of considerations. The advantages of simulated annealing are its simplicity of implementation, ability to incorporate constraints, and applicability to any computable cost function on trees. The disadvantage is the computer run time requirement as problem size increases.

REFERENCES

[1] E. AARTS and J. KORST, 1989, *Simulated Annealing and Boltzmann Machines,* John Wiley and Sons.

[2] A. J. BAYES, 1973, A dynamic programming algorithm to optimize decision table code, *Australian Computer Journal,* **5,** 77-79.

[3] L. BREIMAN, J. FRIEDMAN, R. OLSHEN and C. STONE, 1984, *Classification and Regression Trees,* Wadsworth Int.

[4] R. S. BUCY and R. S. DIESPOSTI, 1991, *Classification Tree Optimization by Simulated Annealing* (The Aerospace Corp., P.O. Box 92957, Los Angeles, California, USA 90009-2957. ATR No 91 (8073)-1.

[5] A. CORANA, M. MARCHESI, C. MARTINI, S. RIDELLA, 1987, Minimizing multimodal functions of continuous variables with the simulated annealing algorithm, *ACM Trans. on Mathematical Software,* **13,** 262-280.

[6] G. DEWEY, 1950, *Relative Frequency of English Speech Sounds*, Harvard University Press.

[7] A. EL GAMAL, L. HEMACHANDRA, I. SHPERLING and V. WEI, 1987, Using simulated annealing to design good codes, *IEEE Trans. on Information Theory* **33,** 116-123.

[8] R. G. GALLAGER, 1968, *Information Theory and Reliable Communication*, John Wiley and Sons.

[9] M. R. GAREY, 1970, *Optimal Binary Decision Trees for Diagnostic Identification Problems,* Ph. D. Thesis, Univ. of Wisconsin.

[10] M. R. GAREY and R. L. GRAHAM, 1974, Performance bounds on the splitting algorithm for binary testing, *Acta Informatica*, **3,** 347-355.

[11] C. R. P. HARTMANN, P. K. VARSHNEY, K. G. MEHROTRA and C. L. GERBERICH, 1982, Application of information theory to the construction of efficient decision trees, *IEEE Trans. on Information Theory*, **28,** 565-577.

[12] J. A. HERRERA-BALL, 1988, *Theoretical Foundations and Algorithms for the Generation of Optimal Decision Trees,* Ph. D. Thesis, Univ. of Tennessee.

[13] S. KIRKPATRICK, C. D. GELATT, Jr. and M. P. VECCHI, 1983, Optimization by simulated annealing, *Science,* **220,** 671-680.

[14] N. METROPOLIS, A. ROSENBLUTH, M. ROSENBLUTH, A. TELLER and E. TELLER, 1953, Equation of state calculation by fast computing machines, *J. of Chem. Physics,* **21,** 1087-1092.

[15] B. MORET, 1982, Decision trees and diagrams, *Computing Surveys* **14,** 593-623.

[16] O. MURPHY, 1990, A unifying framework for trie design heuristics, *Information Processing Letters* **34,** 243-249.

[17] L. T. REINWALD, R. M. SOLAND, 1966, Conversion of limited entry decision tables to optimal computer programs I : minimum average processing time, *J. Ass. Comput. Mach.* **13,** 339-358.