DIMACS Technical Report 2006-05

Experimental Analysis of Sequential Decision Making Algorithms for Port of Entry Inspection Procedures

by

Saket Anand, David Madigan, Richard Mammone,
Saumitr Pathak, Fred Roberts

Rutgers University
Piscataway, NJ 08854

Supported by ONR grant number N00014-05-1-0237 and NSF grant number NSFSES 05-18543 to Rutgers University. The authors thank Kevin Saeger and Phillip Stroud for providing values of parameters and code to assist us in the analysis in this paper.

Abstract

Following work of Stroud and Saeger, we investigate the formulation of the port of entry inspection algorithm problem as a problem of finding an optimal binary decision tree for an appropriate Boolean decision function. We report on an experimental analysis of the robustness of the conclusions of the Stroud-Saeger analysis and show that the optimal inspection strategy is remarkably insensitive to variations in the parameters needed to apply the Stroud-Saeger method.

1. Introduction

As a stream of containers arrives at a port, a decision maker has to decide how to inspect them, which to subject to further inspection, which to allow to pass through with only minimal levels of inspection, etc. We look at this as a complex sequential decision making problem. Stroud and Saeger [8] have formulated this problem, in an important special case, as a problem of finding an optimal binary decision tree for an appropriate binary decision function. In this paper, we report on experimental analysis of the Stroud-Saeger method that has led us to the conclusion that the optimal inspection strategy is remarkably insensitive to variations in the parameters needed to apply the method.

2. Sequential Diagnosis

Sequential decision problems arise in many areas, including communication networks (testing connectivity, paging cellular customers, sequencing tasks, etc.), manufacturing (testing machines, fault diagnosis, routing customer service calls, etc.), artificial intelligence and computer science (optimal derivation strategies in knowledge bases, best-value satisfying search, coding decision tables, etc.), and medicine (diagnosing patients, sequencing treatments, etc.). A selected list of references for such applications includes [4, 6, 7].

Sequential diagnosis is an old subject, but one that has become increasingly important with the need for new models and algorithms as the traditional methods for making decisions sequentially do not scale.

3. Problem Formulation

The problem we investigate is to find algorithms for sequential diagnosis that minimize the total "cost" of the inspection procedure, including the cost of false positives and false negatives. To make the problem precise, we imagine a stream of containers arriving at the port with the goal of classifying each of them into one of several categories. In the simplest case, these are "ok" (0) or "suspicious" (1). There are several possible tests that can be performed and an inspection scheme specifies which test to perform next based on outcomes of previous tests. We can think of the containers as having certain attributes, such as levels of certain kinds of chemical or biological materials, whether or not certain types of cargo are present in the cargo list, and whether cargo was picked up in a certain port. At present, inspectors use attributes such as:

Does the container's ship's manifest set off an "alarm"? Is the neutron or Gamma emission count above threshold? Does a radiograph image come up positive? Does an induced fission test come up positive? We can imagine many other attributes. Our study is concerned with general algorithmic approaches. We seek a methodology that is not necessarily tied to today's technology. Detectors are evolving quickly and so this approach makes sense to us.

3.1. Boolean Decision Functions and Corresponding Binary Decision Trees

In the simplest case, the attributes can be described as being in one of two states, either 0 ("absent") or 1 ("present"), and we can think of a container as corresponding to a binary string such as 011001. Classification then corresponds to a binary decision function *F* that assigns each binary string to a category. For instance, *F*(011001) = 1 means that we say that a package is suspicious if it has the second, third, and sixth attributes. If the category must be 0 or 1, *F* is a *Boolean decision function* (*BDF*). An inspection scheme tells us in which order to calculate the binary string so as to be able to compute the Boolean function *F*. Stroud and Saeger look at this as the problem of finding an optimal *binary decision tree* (*BDT*) for calculating *F*. In the BDT, the nodes are sensors or categories (0 or 1). Two arcs exit from each sensor node, labeled left and right. Take the right arc when the sensor says the attribute is present, the left arc otherwise. For instance, in Figure 1, we reach category 1 from the root only through the path a_0 to a_1 to 1. A container is classified in category 1 iff it has both attributes a_0 and a_1. The corresponding Boolean function is given by *F*(11) = 1, *F*(10) = *F*(01) = *F*(00) = 0. In Figure 2, we reach category 1 from the root by a_0 left to $a_{1,}$ then right to $a_{2,}$ then right to 1, or a_0 right to a_2 right to 1. A container is classified in category 1 iff it has a_1 and a_2 and not a_0 or a_0 and a_2 and possibly a_1. The corresponding Boolean function is given by *F*(111) = *F*(101) = *F*(011) = 1, F(*abc*) = 0 otherwise. Figure 3 gives a BDT corresponding to the same Boolean function. However, it has one less observation node a_i. So, it is more efficient if we simply count number of observation nodes.

Even if the Boolean function *F* is fixed, the problem of finding the "optimal" BDT for it is very hard (NP-complete) [5]. One can try to solve it by brute force enumeration. However, even if the number of attributes *n* is 4, this is not practical. In present-day practice in the Port of Long Beach/Los Angeles, the nation's busiest port, *n* = 4. Several classes of BDFs have been found for which an efficient solution is possible.

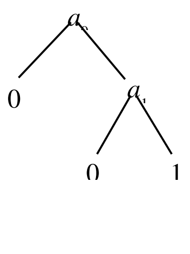

Figure 1.

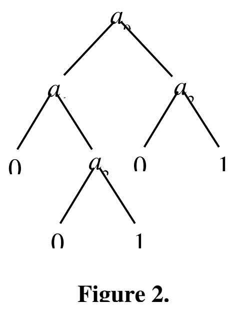

Figure 2.

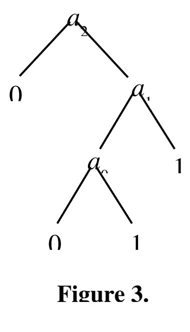

Figure 3.

This is the case for *k*-out-of-*n* systems, certain series-parallel systems, read-once systems, “regular systems”, and Horn systems.

3.2. Complete, Monotone Boolean Functions

One approach to the problem, therefore, is to make special assumptions about the Boolean function *F*. For “monotone” Boolean functions, integer programming formulations give promising heuristics. A Boolean function is *monotone* if given two strings $x_1,x_2\ldots x_n$, $y_1,y_2\ldots y_n$ with $x_i \geq y_i$ for all i, then $F(x_1,x_2\ldots x_n) \geq F(y_1,y_2\ldots y_n)$. Stroud and Saeger limit their analysis to complete, monotone Boolean functions, where a Boolean function F is incomplete if *F* can be calculated by finding at most n-1 attributes and knowing the value of the input string on those attributes. The rationale for limiting the analysis is that sensors detect “bad” things so a positive reading should make things worse (monotonicity) and that all sensors should be in a BDT (completeness). Stroud and Saeger enumerate all complete, monotone Boolean functions and then calculate the least expensive corresponding BDTs under assumptions about various costs associated with the trees. Their method is practical for *n* up to 4, but not $n = 5$.

The problem is exacerbated by the number of BDFs. For example, for $n = 2$, there are 6 monotone Boolean functions; only 2 are complete and monotone; and there are 4 binary decision trees for calculating these 2 complete, monotone Boolean functions. For $n = 3$, there are 9 complete, monotone Boolean functions and 60 distinct binary trees for calculating them. For $n = 4$, there are 114 complete, monotone Boolean functions and 11,808 distinct corresponding BDTs. Compare this with 1,079,779,602 BDTs for all Boolean functions! For $n = 5$, there are 6894 complete, monotone Boolean functions and 263,515,920 corresponding BDTs. Even worse: compare 5 x 10^{18} BDTs corresponding to all Boolean functions. (Counts are from Stroud-Saeger.)

3.3. Cost of a BDF

We seek a least-cost Boolean function and in particular a least-cost corresponding BDT. How does one calculate cost? The cost of an inspection scheme is not just measured by the number of sensors in the BDT. Using a sensor has several costs: the unit cost of inspecting one item with it, the fixed cost of purchasing and deploying it, and the delay cost from queuing up at the sensor station. In our study, we have disregarded the fixed and delay costs and so sought to minimize unit costs. Of course, unit costs should be looked at probabilistically. How many nodes of the decision tree are actually visited during the "average" container’s inspection? This depends on the "distribution" of containers. In this study, we assume this distribution has been used to obtain the probability of sensor errors, we also assume we know the probability of a bomb in a container, and we seek to estimate the expected cost of utilizing a tree, the expected sum of unit costs. We denote this expected utilization cost by C_{util}. More sophisticated models would include models of the distribution of attributes of containers and a more refined analysis of expected cost of utilizing the tree, bringing in delay costs. The other key costs associated with a BDF or corresponding BDT are the cost of a false positive and of a false negative. The former is the cost of additional tests. If it means opening the container, it is relatively expensive. The latter involves complex issues such as estimating the cost of a bomb going off in a large city.

3.4. Sensor Errors

A more refined analysis models sensor errors. In the simplest model, we assume that all sensors checking for attribute a_i have the same fixed probability of saying a_i is 0 if in fact it is 1, and similarly saying it is 1 if in fact it is 0. A more sophisticated analysis later will describe a model for determining probabilities of sensor errors. In what follows, we use the notation X for state of nature (bomb or no bomb) and Y for the outcome 0 or 1 of a sensor inspection or of the entire inspection process. The total (expected) cost of utilizing a tree is given by

$$C_{Tot} = C_{FP} * P_{FP} + C_{FN} * P_{FN} + C_{util} \qquad \textbf{(0)}$$

where C_{FP} is the cost of false positive (Type I error); C_{FN} is the cost of false negative (Type II error); P_{FP} is the probability of a false positive occurring; P_{FN} is the probability of a false negative occurring; C_{util} is the expected cost of utilization of the tree.

4. The Stroud-Saeger Calculations

Stroud and Saeger ranked all trees formed from three or four sensors according to increasing tree costs. We denote sensors by A, B, C and D. Stroud and Saeger used the cost function defined in Equation (1). Assumptions in their work were specific values for the properties of the sensors, i.e., cost of utilizing them and probabilities of false positive and false negative sensor outcomes. Specifically, the values used in their analysis were as follows, where C_A is the unit cost of utilizing a sensor of type A, Y_A the outcome of inspection on a container by sensor A, etc.

$C_A = .25$	$P(Y_A=1\|X=1) = .9856$	$P(Y_A=1\|X=0) = .0144$
$C_B = 10$	$P(Y_B=1\|X=1) = .7779$	$P(Y_B=1\|X=0) = .2221$

$C_C = 30$ $\quad$ $P(Y_C=1|X=1) = .9265$ $\quad$ $P(Y_C=1|X=0) = .0735$

$C_D = 1$ $\quad$ $P(Y_D=1|X=1) = .9893$ $\quad$ $P(Y_D=1|X=0) = .0107$

Also fixed were the three parameters we call ***base parameters***: C_{FN}, C_{FP}, P(X=1). The purpose of our work was to explore the sensitivity of the Stroud-Saeger conclusions about optimal BDTs to changes in values of the parameters defining the problem. In this paper, we explore changes in the base parameters.

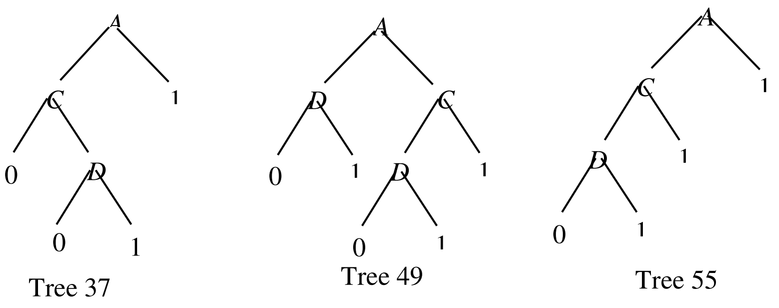

Figure 4. Trees that attained top rank for experiments with n = 3.

5. Sensitivity Analysis for the Case of Three Attributes

We started by looking at the case n = 3 and used sensors A, C and D in our BDTs. In our computer experiments, we used ranges for the values of base parameters. C_{FN} was varied between \$25 million and \$500 billion. These are low and high estimates of the direct and indirect costs incurred due to a false negative - interpreted as the cost of a bomb going off in a large city. Stroud and Saeger used the value \$50 billion and we sought to use numbers that were much higher and much lower than this. C_{FP} was varied between \$180 and \$720. This is interpreted as the cost incurred due to a false positive, which requires opening the container with 4 men working on it. The estimates ranged from a low of 4 men working 3 hours at a salary of \$15/hour to 4 men working 6 hours at a salary of \$30/hour. Stroud and Saeger used the value \$600. Finally, P(X=1) was varied between $3x10^{-9}$ and $1x10^{-5}$. These numbers were chosen to give a wide range around the Stroud-Saeger-assumed value of $3x10^{-8}$.

In the first sets of experiments, we chose a fixed value for one of the base parameters. In one set of experiments, the value was chosen at random from its interval, and in another, the value was fixed at that used by Stroud and Saeger. In each of 10,000 runs, we picked the values of the other two base parameters randomly and uniformly from their interval of values and then found the highest ranked tree. Results of the experiments are shown in Table 1.

In all of these experiments, only three trees out of a possible 60 ever came out ranked first: Trees numbered 37, 49, and 55 in the Stroud-Saeger enumeration. They are shown in Figure 4. Tree 55 was predominantly the top-ranked tree, except in the runs when P(X=1) was fixed at the Stroud-Saeger value, in which case tree 37 was predominantly first. Tree 49 never appeared more than 1.21% of the time in any one of the experiments.

Similar experiments were performed by fixing two of the base parameters and varying the third, with fixed values either being chosen (independently) randomly in their intervals or at the Stroud-Saeger values. Results are shown in Table 2. Again, only the same three trees, 37, 49, and 55, were ever ranked first, with trees 37 and 55 again dominating and tree 49 only appearing once more than 1% of the time. The robustness of the results of the experiments with $n = 3$ is quite surprising.

A comparison of the BDF corresponding to trees 37, 49, and 55 is also interesting. Tree 37 corresponds to the Boolean expression 00011111, which represents the sequence $F(000)F(001)\ldots F(110)F(111)$. Tree 49 corresponds to the Boolean expression

Table 1 Frequency that trees attained top rank when n = 3. One parameter was fixed at a randomly selected value from its interval and then at Stroud and Saeger values, and the other two parameters were assigned 10,000 randomly chosen values from their intervals

Variables	**Fixed**	**Randomly Selected Values**			**Stroud and Saeger Values**		
		Value	**Tree No.**	**Frequency**	**Value**	**Tree No.**	**Frequency**
P(X=1)	C_{FN}	$8.2737\text{x}10^{10}$	37	343	$5\text{x}10^{10}$	37	441
C_{FP}			49	37		49	66
			55	9620		55	9493
C_{FN}	P(X=1)	$0.5538\text{x}10^{-5}$	37	99	$3\text{x}10^{-8}$	37	9923
C_{FP}			49	8		55	77
			55	9893			
P(X=1)	C_{FP}	668.1793	37	412	600	37	541
C_{FN}			49	121		49	53
			55	9467		55	9406

Table 2 Frequency that trees attained top rank when n = 3. Two parameters were fixed at randomly selected values from their intervals and then at Stroud and Saeger values, and the third parameter was assigned 10,000 randomly chosen values from its interval

Variables	**Fixed**	**Randomly Selected Values**			**Stroud and Saeger Values**		
		Value	**Tree No.**	**Frequency**	**Value**	**Tree No.**	**Frequency**
C_{FN}	P(X=1)	$0.1281x10^{-5}$	37	568	$3x10^{-8}$	37	10000
	C_{FP}	492.6116	55	9432	600		
P(X=1)	C_{FN}	$4.747x10^{11}$	37	54	$5x10^{10}$	37	694
	C_{FP}	351.9526	55	9946	600	49	108
						55	9198
C_{FP}	P(X=1)	$0.8373x10^{-5}$	55	10000	$3x10^{-8}$	37	10000
	C_{FN}	$4.2681x10^{11}$			$5x10^{10}$		

01010111 and tree 55 to the Boolean expression 01111111. Thus, in Tree 55, a container is called suspicious if it fails at least one test. In Tree 37, a container is called suspicious if it fails the first test or if it fails both the remaining tests. In Tree 49, a container is called suspicious if it fails at least two tests or fails only the third one.

6. Sensitivity Analysis for the Case of Four Attributes

Turning to the case n = 4, we used sensors A, B, C, and D in our BDTs and used the same interval of values for the parameters C_{FP}, C_{FN} and P(X=1) as before. We ran the same kinds of experiments as in the case n = 3. Tables 3 and 4 show the results (omitting trees that were rarely ranked first). When one base parameter was fixed, only five trees ever appeared first more than 1% of the time in an experiment: Trees numbered 6797, 8965, 9133, 11605, and 11785 in the Stroud-Saeger numbering. Each appeared first at least 5% of the time in at least one experiment. These trees are shown in Figure 5. If we consider trees appearing first at least .99% of the time, only tree 11305 gets added to the list. In each experiment, one tree dominated first place, except in the experiment where P(X=1) was fixed at the Stroud-Saeger value, when two trees dominated. Considering the fact that there are 11,808 candidate trees, this is remarkable stability of results.

In the case where two base parameters were held fixed, again the same five trees dominated as the only trees appearing first at least 1% of the time in any experiment. Indeed, only they appeared first at least .02% of the time in any experiment. Only a sixth tree, 11305, which also appeared in the experiments with one base parameter held fixed, appeared if we consider trees that were ranked first in at least .017% of the runs in some experiment. Again, the stability of the top-ranked trees is quite striking.

For the five top trees, it is interesting the compare the Boolean expression corresponding to $F(0000)F(0001)\ldots F(1011)F(1111)$. Tree 6797 corresponds to expression

Table 3 Frequency that trees attained top rank when n = 4. One parameter was fixed at a randomly selected value from its interval and then at Stroud and Saeger values, and the other two parameters were assigned 10,000 randomly chosen values from their intervals. Trees with small frequency of top rank are not shown

Variables	**Fixed**	**Randomly Selected Values**			**Stroud and Saeger Values**		
		Value	**Tree No.**	**Frequency**	**Value**	**Tree No.**	**Frequency**
P(X=1)	C_{FN}	$4.7485\text{x}10^{10}$	8965	121	$5\text{x}10^{10}$	8965	117
C_{FP}			9133	392		9133	374
			11305	99		11305	96
			11605	9351		11605	9381
C_{FN}	P(X=1)	$0.6344\text{x}10^{-5}$	9133	65	$3\text{x}10^{-8}$	6797	558
C_{FP}			11605	7928		8965	3833
			11785	1979		9133	5406
P(X=1)	C_{FP}	453.6849	9133	235	600	9133	184
C_{FN}			11605	8621		11605	9232
			11785	1062		11785	333

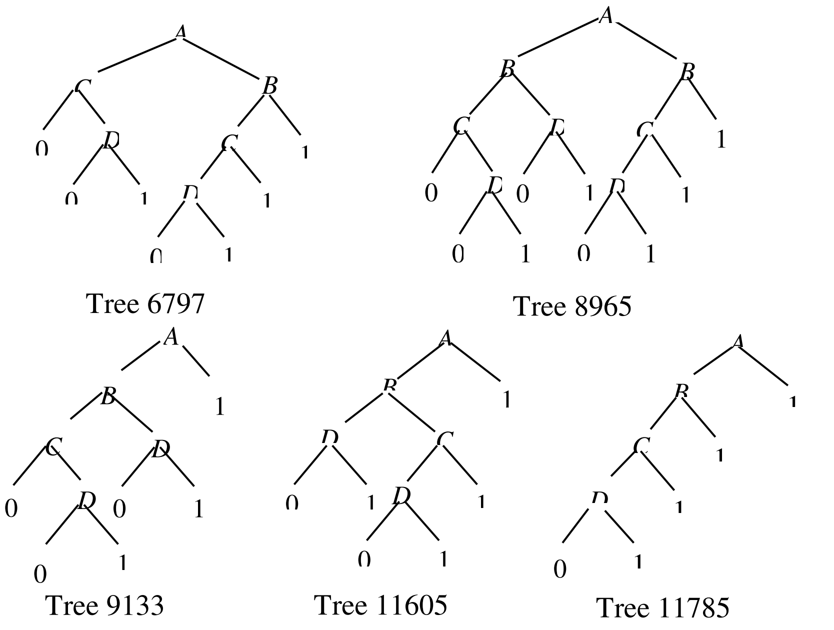

Figure 5. Trees that attained top rank most frequently for the experiments with n = 4

Table 4 Frequency that trees attained top rank when n = 4. Two parameters were fixed at randomly selected values from their intervals and then at Stroud and Saeger values, and the third parameter was assigned 10,000 randomly chosen values from its interval. Trees with small frequency of attaining top rank for n = 4 are not shown

Variables	**Fixed**	**Randomly Selected Values**			**Stroud and Saeger Values**		
		Value	**Tree No.**	**Frequency**	**Value**	**Tree No.**	**Frequency**
C_{FN}	P(X=1)	0.6284×10^{-5}	9133	47	3×10^{-8}	6797	897
			11605	3614		8965	8671
	C_{FP}	188.5681	11785	6339	600	9133	309
P(X=1)	C_{FN}	4.0624×10^{11}	9133	44	5×10^{10}	8965	237
			11605	4551		9133	357
	C_{FP}	297.5277	11785	5405	600	11305	170
						11605	9156
C_{FP}	P(X=1)	0.5992×10^{-5}	11605	9087	3×10^{-8}	6797	2336
	C_{FN}	2.4041×10^{11}	11785	913	5×10^{10}	8965	3942
						9133	3722

0001000101111111, tree 8965 to 0001010101111111, tree 9133 to 0001010111111111, tree 11605 to 0101011111111111, and tree 11785 to

0111111111111111. The expressions for trees 11605 and 11785 differ in only two places, as do those for trees 9133 and 11605. There can be quite a difference in Boolean expressions, however. Those for trees 6797 and 11785 differ in six places.

7. Modeling Sensor Errors using Thresholds

One approach to sensor errors involves modeling sensor operation/interpretation of sensor readings. A natural model, one used by Stroud and Saeger, is a threshold model using counts (e.g., Gamma radiation counts). If the count exceeds some threshold, we conclude that the attribute being tested for is present. To describe such a threshold model, suppose that sensor i has discriminating power K_i and let the threshold for sensor i be denoted by T_i. We calculate the fraction of containers in each category whose readings exceed the threshold. While sensor characteristics are a function of design and environmental conditions, the thresholds can be set by the decision maker.

The Stroud-Saeger approach is to seek threshold values that minimize all costs: inspection, false positive/negative. The readings of sensors are also determined by their design and environmental conditions. Let us assume that readings of category 0 containers (those not containing a bomb) follow a Gaussian distribution and similarly category 1 containers (those containing a bomb). See Figure 6, in which Σ_i is the relative spread factor and P_D is the probability of detection by the i^{th} sensor $\text{P}(Yi{=}1|X{=}1)$ while P_F is the probability of a false positive at the ith sensor $\text{P}(Yi{=}1|X{=}0)$. The probability of false positive for the i^{th} sensor is computed as:

$$P(Y_i{=}1|X{=}0) = 0.5\ erfc[T_i/\surd 2], \qquad (\mathbf{0})$$

while the probability of detection for the ith sensor is computed as

$$P(Y_i{=}1|X{=}1) = 0.5\ erfc[(T_i\text{-}K_i)/(\Sigma\surd 2)], \qquad (\mathbf{0})$$

where $erfc$, the complementary error function, is given by

$$erfc(x) = \Gamma(½,x^2)/\surd\ \pi. \qquad (\mathbf{0})$$

We ran experiments with this model by choosing the following values of sensor parameters also used by Stroud-Saeger: $K_A = 4.37$, $\Sigma_A = 1$; $K_B = 1.53$, $\Sigma_B = 1$; $K_C = 2.9$, $\Sigma_C = 1$; $K_D = 4.6$, $\Sigma_D = 1$. We then varied the individual sensor thresholds T_A, T_C and T_D (for $n = 3$) from -4.0 to +4.0 in steps of 0.4. These values were chosen since they gave us an “ROC curve” (see Section 8) for the individual sensors over a complete range $\text{P}(Y_i{=}1|X{=}0)$ and $\text{P}(Y_i{=}1|X{=}1)$. The base parameters were chosen at randomly selected values in these intervals.

In the case $n = 3$, 68,921 experiments were conducted, as each T_i was varied through its entire range. As seen from Table 5, a total of 15 different trees were ranked first in these experiments, more than were obtained in our earlier experiments. Tree 37 had the highest frequency of attaining rank one, appearing first 17,515 times. A few of the other trees that ranked first a relatively large number of times were trees numbered 7 and 55. Note that 37 and 55 also predominated in our other experiments.

In the case $n = 4$, 194,481 similar experiments were conducted. A total of 244 trees ranked first, with tree 445 the most frequent (13,012 times). Other trees often ranked first in our other experiments were here too: 9133, 11605, and 11785. Results for these experiments with $n = 4$ are summarized in Table 5.

8. Using the ROC Curve

We can use Receiver Operating Characteristic (ROC) curves to identify optimal thresholds for sensors. The ROC curve is the plot of the probability of correct detection (P_D) vs. the probability of a false positive (P_F). The ROC curve is used to select an operating point, which provides the tradeoff between P_D and P_F. Each sensor has a ROC curve and the combination of the sensors into a decision tree has a composite ROC curve. We seek operating characteristics of sensors that place us in the upper left hand corner of the ROC curve. Here, P_F is small and P_D is large.

The parameter that is varied to get different operating points on the ROC curve is the sensor threshold and a combination of thresholds for the decision tree. The Equal Error Rate (EER) is the operating point on the ROC curve where $P_F = 1 - P_D$. By studying the performance characteristics (P$(Y=1|X=0)$, P$(Y=1|X=1)$) of the tree over

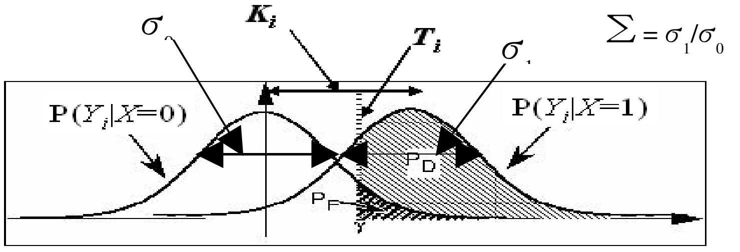

Figure 6. Typical sensor characteristics.

all combinations of sensor thresholds (T_i) and studying the region of high detection probabilities and low false positive probabilities, we can use the ROC curve to choose threshold values in practice. Assuming performance probabilities (P$(Y=1|X=1)$ and P$(Y=1|X=0)$) to be monotonically related (in the sense that P$(Y=1|X=1)$ can be called a monotonic function of P$(Y=1|X=0)$), we can find an ROC curve for the tree consisting of the set containing maximum P$(Y=1|X=1)$ value corresponding to given P$(Y=1|X=0)$ value.

9. Conclusions, Discussion, and Further Work

Our work has shown a remarkable robustness in the conclusions about optimal binary decision trees in sequential decision making applications in inspection applications. Very few trees arise as optimal over a wide range of choices of values for the key parameters in the model. Moreover, there is also considerable robustness in the optimal Boolean decision function – very few decision functions correspond to the optimal trees and those that do often are closely related. We do not yet have a good theoretical explanation for these conclusions about robustness. We also do not have a good understanding, as yet, of the relations between the different optimal trees, in particular their tree structure. Our results should be regarded as preliminary, though intriguing and suggesting many questions.

Table 5 Frequency that trees attained top rank when thresholds were varied. The base parameters were fixed at randomly selected values from their intervals. For n = 3 sensors, 15 out of 60 trees came out to be top rank. For *n* = 4 sensors, only 244 out of 11,808 trees attained top rank. Trees with small frequency of attaining top rank for *n* = 4 are not shown

***n*=3, number of experiments = 68,921**			***n*=4, number of experiments = 194481**		
Constants	**Tree No.**	**Frequency**	**Constants**	**Tree No.**	**Frequency**
C_{FN} = $5.0125\text{x}10^9$ P(X=1) = $5.05\text{x}10^{-6}$ and C_{FP} = 450	1	5828	C_{FN} = $4.8668\text{x}10^{11}$ P(X=1) = $7.5361\text{x}10^{-6}$ and C_{FP} = 499.75	87	3402
	2	183		145	11143
	7	13392		325	5574
	15	2437		386	4018
	19	5256		445	13012
	23	1475		505	10545
	25	957		506	5249
	27	114		2617	10139
	29	146		5761	5942
	37	17515		8003	4539
	38	4572		9133	9280
	45	5873		10783	4496
	49	264		11605	10958
	51	322		11785	5910
	55	10587		11791	5196

As Stroud and Saeger have noted, their method does not scale very well. We have already reached limits of computation for the case of *n* = 4 types of inspections. With *n* = 5, similar experiments seem infeasible. We did these calculations with Matlab on a Pentium IV 3 GHz processor, with 1GB of RAM. Other methods are needed to find optimal trees if we have more types of sensors and to investigate the sensitivity of the conclusions.

There are more experiments that we propose to do to test sensitivity of the Stroud-Saeger conclusions. In particular, our experiments have fixed the characteristics of the sensors through such parameters as C_A, K_A, $P(Y_A = 1|X = 0)$, etc. We only did experiments by using specific values of parameters. The optimal tree is certainly related to these characteristics of the available sensors. We will do other experiments in which these values are varied.

We have not done a lot of analysis of the robustness of the second, third, and fourth-ranked trees, though initial results show a great deal of robustness akin to that reported here. In practical applications, a near-optimal tree might very well be a perfectly acceptable solution to the inspection problem. In this case, there might be more efficient methods for finding near-optimal trees than the brute force methods described by Stroud and Saeger and tested here.

The methods here and in Stroud-Saeger depend heavily on being able to limit the number of possible Boolean decision functions and hence the number of possible BDTs. In particular, they depend on the monotonicity and completeness assumptions. More work is needed to explore alternative assumptions that are relevant to the port of entry inspection applications.

In practice, one thinks of n types of sensors that measure presence or absence of the n attributes. There are many copies of each sensor. A complication is that different sensor types have different characteristics. As containers arrive for inspection, we have to decide which sensor of a given type to use. The containers are sent to different inspection lanes, each having a particular test (sensor), and the containers form queues. Recall that the "cost" of inspection includes the cost of failure, including failure to foil a terrorist plot. There are many ways to lower the total "cost" of inspection. Only one is to use more efficient orders of inspection. Others are to find ways to inspect more containers, to find ways to cut down on delays at inspection lanes, etc. More complicated cost models would bring in costs of delays and also consider the limits on delays that are imposed by the need to keep the port operating. Besides efficient inspection schemes, one could decrease costs by buying more sensors or changing the allocation of containers to sensor lanes.

Another variant of the Stroud-Saeger model would have us go to more than two values of an attribute, e.g., present, absent, present with probability > 75%, absent with probability at least 75%; or ok, not ok, ok with probability > 99%, ok with probability between 95% and 99%. Still another approach would have us infer the Boolean function from observations. There is a considerable literature that deals with partially defined Boolean functions (see for example [1, 2, 3]). Still another approach would use machine learning methods to learn the thresholds of the sensors in order to minimize the misclassification error of the entire tree, subject to the constraint of minimizing the total cost of the generated tree.

References

1. Boros, E., Ibaraki, T. and Makino, K., "*Logical Analysis of Binary Data with Missing Bits,*" Artificial Intelligence, 107 (1999), 219-263.
2. Boros, E., Ibaraki, T. and Makino, K., "*Variations on Extending Partially Defined Boolean Functions with Missing Bits,*" Information and Computation, 180 (2003), 53-70.
3. Chiu, S.Y., Cox, L.A. and Sun, X., "*Least-Cost Failure Diagnosis in Uncertain Reliability Systems,*" Reliability Engineering and System Safety, 54 (1996), 203-216.
4. Duffuaa, S. O., and Raouf, A., "*An Optimal Sequence in Multicharacteristics Sequence,*" Journal of Optimization Theory and Applications, 67 (1990), 79-87.
5. Hyafil, L. and Rivest, R. L., "*Constructing Optimal Binary Decision Trees is NP-Complete,*" Information Processing Letters, 5 (1976), 15-17.
6. Lauritzen, S. N., and Nilsson, D., "*Representing and Solving Decision Problems with Limited Information,*" Management Science, 47 (2001), 1238-1251.
7. Simon, H. A. and Kadane, J. B., "*Optimal Problem-Solving Search: All-or-None Solutions,*" Artificial Intelligence, 6 (1975), 235-247.
8. Stroud, P. D. and Saeger K. J., "*Enumeration of Increasing Boolean Expressions and Alternative Digraph Implementations for Diagnostic Applications,*" Proceedings Volume IV, Computer, Communication and Control Technologies, (2003), 328-333