RUTCOR
RESEARCH
REPORT

# Large Scale LP Model for Finding Optimal Container Inspection Strategies

E. Boros[a] L. Fedzhora[b] P. B. Kantor[c]
K. Saeger[d] P. Stroud[e]

RRR 26-2006, October, 2006

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003; boros@rutcor.rutgers.edu
[b]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003; fedzhora@rutcor.rutgers.edu
[c]SCILS, Rutgers University; paul.kantor@rutgers.edu
[d]LANL; saeger@lanl.gov
[e]LANL; stroud@lanl.gov

Rutcor Research Report
RRR 26-2006, October, 2006

# Large Scale LP Model for Finding Optimal Container Inspection Strategies

E. Boros L. Fedzhora P. B. Kantor K. Saeger P. Stroud

**Abstract.** Cargo ships arriving at USA ports are inspected for unauthorized materials. Ideally, we want to open and check every container they carry, but it is costly and time-consuming. Instead, tests are developed to decide whether a container should be opened. By utilizing a polyhedral description of decision trees, we develop a large scale linear programming model for sequential container inspection that determines an optimal inspection strategy under various limitations, improving on earlier approaches in several ways: (a) we consider mixed strategies and multiple thresholds for each sensor, which provide more effective inspection strategies; (b) our model can accommodate realistic limitations (budget, sensor capacity, time limits, etc.), as well as multiple container types; (c) our model is computationally more tractable allowing us to solve cases that were prohibitive in preceding models, and making it possible to analyze the impact of potentially new sensor technologies.

**Acknowledgements**: The authors are thankful for the partial support by the National Science Foundation (Grant NSFSES 05-18543) and the Office of Naval Research (Grant N00014-05-1-0237).

# 1 Introduction

We consider the problem of deploying a set of sensors most effectively when the goal is to detect as many as possible of some extremely rare contraband, which we refer to here as "bad" contents. The sensors have operating costs (part of which is proportional to the number of containers inspected, and part of which is independent of the inspected containers), and they have individual capacity limitations (which can be extended in discrete steps at certain investment costs). There is also a "manual checking" procedure, $CHK$, which is considered 100% effective, and considerably more expensive than the application of other sensor technologies. As is widely known (see e.g., newspaper articles) security at large US ports does not even consider applying sensors to all containers. Rather, most containers are cleared on the basis of other examinations (e.g., some sort of inspection done at the port of origination). Hence we expect the methods presented here to be applied only to the 5-10 percent of all containers that are judged to need further inspection.

Whether the specific sensors are hand-held and portable, or fixed installations through which the containers must be moved, a container may be routed through the sensors according to the readings it has produced. This can be conceptualized by saying that a particular design of the inspection schedule produces a tree. The first sensor to be applied is at the root node. According to the results on that sensor, the container may be released, checked manually, or sent to another sensor, etc. Stroud and Saeger [4] have formulated the problem in precisely this way. They postulate that each sensor makes a binary decision, which can be thought of as "*more suspicious*" or "*less suspicious*", and they note that the number of possible binary trees is doubly exponential in the number of sensors. They also note that the threshold dividing "*more suspicious*" from "*less suspicious*" need not be fixed in advance, but could be adjusted after the topology of the tree has been set. Any given sensor may appear in different parts of the tree, with, in principle, different threshold settings. They examined the problem, using the same settings for each sensor, wherever it appears. The problem is still a non-linear optimization problem. In that model, costs were assigned to false negatives and false positives, and the combined cost of sensor operations, plus the expected cost of the outcome is minimized by adjusting the thresholds for each tree.

We propose here a number of changes in viewpoint, which, taken together, lead to a more tractable problem, with greater flexibility to incorporate additional constraints/objectives, greater potential for generalization to more sensors, and with some attractive operational advantages. The changes in perspective are the following:

First, we allow that when a sensor is used in different locations in the tree, it may have different thresholds.

Second, we allow the use of more than one threshold at a given sensor (in a given location in a tree) so that the branching following the application of a sensor may be, in general manifold. We refer to this discretization of sensor readings as the "partition" of the outcomes.

Third, we view the inspection process not with a "bird's- eye" view which sees the entire tree, but with a "path- wise" view, which considers the possible histories that any container might experience, under a given overall set of partitions of the readings for each of the

sensors.

Fourth, we reformulate the problem into a comparable form which is more robust in the presence of unknown costs. Specifically, we seek to maximize the detection rate, given constraints on operating budget, capacities, time, etc. We feel that this formulation will be more useful to policy-makers, since they can examine the increase in detection as a function of the increase in budget, and use their own cost coefficients to complete the decision process.

Fifth, we recognize that in the presence of even a single (budgetary, for example) constraint, the optimal detection rate is not necessarily achieved by a pure strategy. When there is a single linear constraint, and a convex problem, the optimum will involve a mixed strategy using at most two pure strategies. As more constraints are added the number of components in the optimal mix rises. Let us remark here that any given decision tree has some vulnerabilities. Namely, assuming intelligent adversaries, the packaging and timing of "bad" content may be adapted to the particular inspection sequence, so that to minimize detection. A mixed strategy is much less vulnerable in this sense, since our adversaries can never be sure in advance about the type of inspection strategy that will be applied[1].

Somewhat surprisingly, complexifying the problem in several ways (moving from trees to paths, considering a larger number of static thresholds rather than a small number of variable ones, seeking mixture solutions directly, etc.) makes the problem of sensor sequencing more tractable, rather than less[2]. First of all, the "pathwise" view of the problem decreases the number of decision variables by an order of magnitude. Secondly, in moving from a problem with adjustable thresholds at each sensor, to a problem with fixed partitions at each sensor, we have replaced a really huge number (the number of trees) of small non-linear problems (finding the thresholds) by a single linear optimization problem with a very large (the number of paths) variables, which is still tractable by standard off-the-shelf software packages. We illustrate the approach with examination of the model discussed in [4], and discuss the effects of choosing different partitions.

To sum up: we formulate a new problem in which each sensor produces a reading which is "quantized" into one of many bins; we describe the problem not in terms of trees, but in terms of paths, which a container may follow through the tree; we exhibit the need for mixed strategies and develop a tractable linear programming formulation.

The mathematical basis for our model is a polyhedral description of all decision trees in the space of paths.

We are able to show that this method, which retains and makes use of more specific information gathered at earlier sensors, is able to provide better results at any given budget. We briefly examine the effects of refining the partitions on both computational cost and overall quality of the solution found.

---

[1] Of course, bribing the harbor master remains an ancient method of proven effectiveness.

[2] "*I can't cure your cold, but if you get pneumonia, I can help you*". (Joke dating to the discovery of penicillin.)

## 2 Detailed Discussion and Background

An inspection strategy involves the sequential application of sensors, where the choice of a next sensor depends on the results obtained in the previous steps. Such an inspection sequence terminates with a final decision about the container. In our study we assume two possible final decisions, either by declaring a container safe to leave the port and enter the country ("OK"), or by subjecting it to a manual inspection ("CHK").

Such inspection strategies customarily are represented as so called *decision trees* (see Section 4 for details). Even if we allow only two different decision to be made after each sensor reading (in other words, if the results of an application of a sensor translated into two classes, e.g., "suspicious" and "not-suspicious"), the number of possible (labeled binary) decision trees is $2^{2^n}$, doubly exponential in the number sensors. Even the number of those decision trees which represent monotone Boolean functions is doubly exponentially large. A complete enumeration of the possible trees, in order to select a "best one" is a computational challenge even for a small number of sensors (see [4]).

The problem of scheduling sensors, and of quantizing the information obtained at the sensors has a long history. Early development was driven by aerospace applications much of it related to the problem of missile detection (see e.g., Varshney [1, 10], Kushner and Pacut [9], Cherikh [2]). Related problems also arise in the use of medical tests (see e.g., Greiner [7]).

We have not found any authors who treated the problem using the approach described here. However, in those papers, particularly in Kushner and Pacut [9] attention is drawn to the fact that what is essential for solving problems of this type is to represent each sensor in terms of its "ROC" performance curve (Radar Operational Curve). Our approach utilizes information about each sensor, which implicitly includes its ROC curve, and so in effect we use the ROC curve, though not in an explicit way.

In our study we use several operational characteristics to describe the effectiveness of a particular inspection strategy, including the *detection rate* (the probability that a truly dangerous container content will be discovered), *inspection cost* (per container average inspection cost), *sensor load* (expected fraction of containers inspected by a particular sensor; related to capacity planning/limitations), *inspection time* (expected time to inspect a container), etc. Ideally, we would like to check manually every container entering the country. However, budget, capacity and time delay limits make this impossible. The application of sensors, which typically can be applied faster and more cheaply than manual inspection, allows for filtering only the truly suspicious containers, which then can feasibly be checked manually. However, since no sensor is perfect, and each provide only a limited amount of information about the content of the inspected containers, such a non-ideal inspection strategy does not guarantee the detection of all dangerous contents.

It is therefore important to understand how high a detection rate one can achieve from a given inspection budget, how this achievable maximum detection rate changes if we change the budget, how much sensor capacity we need, how much delay this causes in the container flow, etc. Furthermore, for planning purposes one must find the best way to build/expand

existing inspection stations, study the effects on efficiency and cost of the introduction of possible new sensor technologies, etc.

To address these and similar questions, we propose a large scale linear programming model, which provides a computationally efficient tool for analysis. The theoretical basis for this model is a polyhedral description of the set of decision trees, in a high dimensional space. Based on this description we show that the above questions can be answered efficiently by linear programming techniques.

Let us point out two particularly important outcomes, in our view, of our study.

The first is the realization that a "best inspection strategy" is not simply a decision tree, but rather an ensemble of decision trees, each applied to a certain fraction of the containers (which decision trees and what fraction will be determined by the optimization model). Note that such a mixed strategy is also much more secure, in the sense that it is much harder to adapt to it by packing methods that trick the inspection strategy.

The second outcome is perhaps more natural. We show that the more information we retain from a sensor reading, the higher the efficiency we can achieve. In other words, using the same set of sensors, the same capacities, and the same inspection budget, one can achieve a higher detection rate (and sometimes substantially higher) by allowing different routing decisions based on the actual readings of previous sensors, rather than coding every sensor reading into a binary decision (e.g., "suspicious" and "non-suspicious").

Many problems remain open, however. One of these is to decide how to define the partitions. Since we are, in effect, making a continuous problem discrete, good choice of the partitions will move the polyhedral description closer to the underlying optimum. There is also the possibility that the mixture problem, for a relatively small number of sensors, can be solved directly for a relatively small number of cases. The solution path here is to seek the optimal decision, at later sensors, based on the precise values generated at the earlier sensors. If, as we are assuming in this paper, the variation in sensors is distributed independently, then for $N$ sensors there are not too many different sequences. In general, there are $N!/k!$ sequences involving $N-k$ of the sensors, and thus in all there are at most $eN!$ basic sequences. However, the determination of the cost- performance curve for each of these is a subtle problem in non-linear optimization, which will be discussed elsewhere.

## 2.1 Sensor Model

Currently available sensors provide only partial and noisy information about the contents of a container. Typical output is either a real number, or an image, which then is translated into a real scale by a human operator.

The usual physical sensor model therefore involves a distribution function, the distribution of sensor readings on a real scale. Of course, for different container types (different container content, container material, container size, etc.) we may have different distributions corresponding to each sensor. We assume that these distributions are known.

For simplicity of presentation, we assume that there are only two types of containers arriving to the port, the *good* ones containing legal content, and the *bad* ones containing dangerous illegal content. This assumption is not necessary for our model, but makes the description simpler.

Let us denote the set of reals by $\mathbb{R}$ (scale of sensor readings), the set of different sensors by $\mathcal{S}$ (typically $|\mathcal{S}|$ is small, say $|\mathcal{S}| = 4$), the set of container types by $\mathcal{T}$ (we could consider several different types of containers; following [4], in this study we consider only two types $\mathcal{T} = \{\text{“good”}, \text{“bad”}\}$), and let $f_{s,\tau}(x)$ denote the density function of sensor readings on sensor $s \in \mathcal{S}$ for containers of type $\tau \in \mathcal{T}$. For a container $C$ and sensor $s \in \mathcal{S}$ let us denote by $R_s(C) \in \mathbb{R}$ the result obtained by applying sensor $s$ to container $C$. That is $R_s(C)$ is the reading of sensor $s$ on container $C$. We assume, following [4] that the same reading will be obtained if the container is remeasured. The realistic background for this assumption is that the primary source of randomness in practice is the large variety of container materials, contents, and varying backgrounds, rather than the uncertainties of the actual measurement.

We also assume, following [4] that for each sensor and each container type we know the distribution of sensor readings, that is the density functions $f_{s,\tau}$ for $s \in \mathcal{S}$ and $\tau \in \mathcal{T}$. Given a subset $X \subseteq \mathbb{R}$ of the reals, we denote by

$$\xi(X, s, \tau) \;=\; \int_X f_{s,\tau}(x)dx \tag{1}$$

the expected fraction of containers of type $\tau \in \mathcal{T}$ which will have a reading in the given range $X$ on sensor $s \in \mathcal{S}$. In other words, $\xi(X, s, \tau)$ is the probability that $R_s(C) \in X$ when $C$ is a randomly selected container of type $\tau$. Given any partition $X_1 \cup X_2 \cup \cdots \cup X_k = \mathbb{R}$ of the set of reals, we clearly have

$$\sum_{i=1}^{k} \xi(X_i, s, \tau) \;=\; 1$$

for every sensor $s \in \mathcal{S}$ and container type $\tau \in \mathcal{T}$.

We consider the proportions $\xi(X_i, s, \tau)$ for certain partitions $\mathbf{P}_s = \{X_1, \cdots, X_{k(s)}\}$, for all sensors $s \in \mathcal{S}$ and container types $\tau \in \mathcal{T}$ as inputs for our model.

Let us add that the distribution of readings on a sensor $s$ may also depend on the results of previous sensor readings, since due to those readings, the operator of sensor $s$ may evaluate/interpret the measurement differently. In other words, the posterior probability that the container is of type $\tau$ may depend on the earlier readings. We might include this in our model, however such a dependence needs more precise data about the sensors. The data available to us does not include such fine detail, and as a consequence, in our computational study we do not consider such dependence.

To clarify our terminology, let us next consider, as a small example, a sensor $a$ and the distribution of “good” and “bad” container readings as shown in Figure 1.

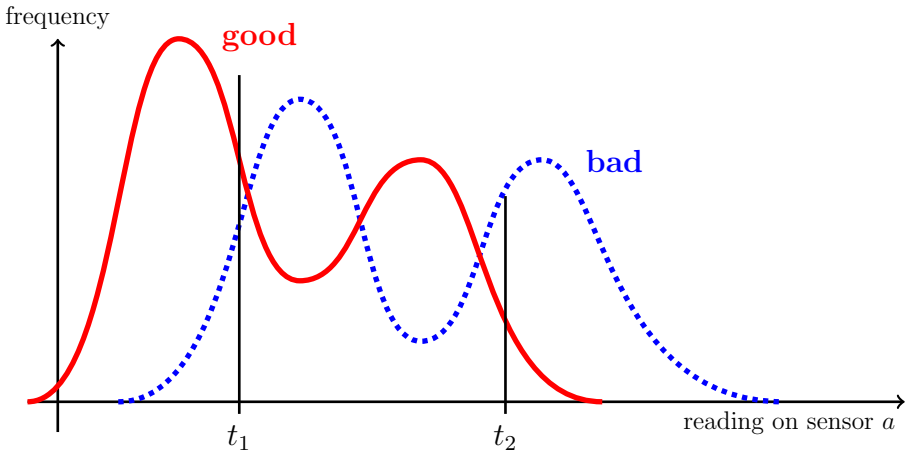


Figure 1: An example for a sensor $a$ and the distributions of sensor readings for "good" and "bad" containers.

Traditionally, sensor readings are discretized by choosing appropriate threshold values. For instance, by choosing two thresholds $t_1$ and $t_2$, we can partition the set of reals into three sets $X_1 = \{x \in \mathbb{R} | x \leq t_1\}$, $X_2 = \{x \in \mathbb{R} | t_1 < x \leq t_2\}$ and $X_3 = \{x \in \mathbb{R} | t_2 < x\}$. Then, we might get

| | $\xi(X_j, a, \tau)$ | |
|---|---|---|
| | $\tau =$ "*good*" | $\tau =$ "*bad*" |
| $j = 1$ | 0.50 | 0.10 |
| $j = 2$ | 0.40 | 0.60 |
| $j = 3$ | 0.10 | 0.30 |

To base some decisions on these readings, we can observe that the odds of having a bad container with a reading on sensor $a$ within range $X_1$ are very low, while they are relatively high in $X_2$ and even higher in $X_3$. This filtering effect can be exploited by applying different sensors for further testing in an inspection strategy, depending on the reading on sensor $a$. Of course, the quality of our decision will strongly depend on which partition we consider and how fine it is.

## 2.2 Pure Inspection Strategies

Let us consider first a small example, involving three sensors $\mathcal{S} = \{a, b, c\}$, and the decision tree $\mathbf{D}$ shown in Figure 2.

This decision tree may represent an inspection strategy, in which we start testing all containers by sensor $a$. Those containers $C$ for which $R_a(C)$ is "*very low*"are deemed to be safe, and let go without any further testing, while those with "*medium*" or "*very high*" readings are deemed suspicious, and tested further. For instance, those with "*very high*" reading on sensor $a$ are tested further by sensor $b$, and if they get a "*high*" reading again,

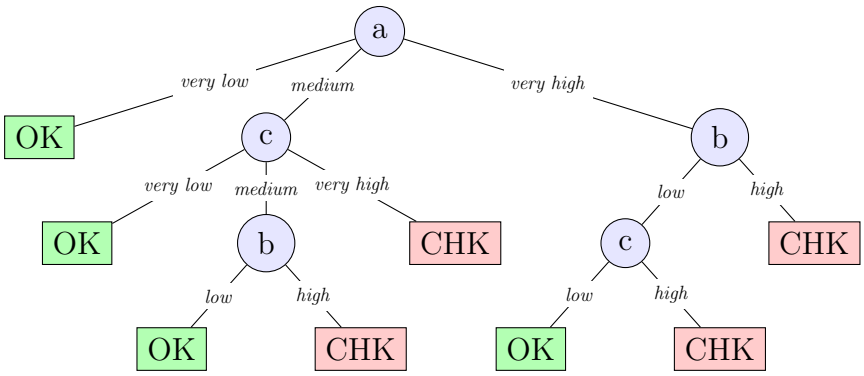


Figure 2: A decision tree utilizing three sensors $\mathcal{S} = \{a, b, c\}$.

then checked manually ("*CHK*"), while those with a "*low*" reading on sensor $b$ are tested even further by sensor $c$, etc.

Let us note that in this example, sensors $c$ and $b$ are used in two different ways. For instance, the readings on sensor $c$ of those containers which have only a "*medium*" previous reading on sensor $a$ are categorized as "*very low*", "*medium*" or "*very high*", while for those containers which had a "*very high*" reading on sensor $a$ and a "*low*" reading on sensor $b$ are categorized simply as "*low*" or "*high*". Furthermore, though the readings on sensor $b$ are always grouped into "*low*" and "*high*", e.g., a "*low*" on $b$ may mean two different things, depending on the previous readings of the containers, as the topology of $\mathbf{D}$ shown in Figure 2 suggests.

Summarizing the above example, we may need seven different thresholds to realize the above decision tree $\mathbf{D}$, two for sensor $a$, two possibly different ones for the two roles of sensor $b$, and three for the two different roles of sensor $c$. Denoting by $\mathbf{t}$ the vector of these thresholds, we view the pair $(\mathbf{D}, \mathbf{t})$ as a *pure inspection strategy*.

In general, once we describe the topology of a decision tree $\mathbf{D}$ and select the necessary thresholds $\mathbf{t}$, the pair $(\mathbf{D}, \mathbf{t})$ describes precisely how to test containers. Furthermore, using the sensor distributions, as we described in the previous subsection, for any pure strategy $(\mathbf{D}, \mathbf{t})$ we can compute the expected fractions of containers of type $\tau \in \mathcal{T}$ which end up in a leaf node labeled "*CHK*", i.e., which are checked manually. Thus, to any pure inspection strategy $(\mathbf{D}, \mathbf{t})$ we can associate several operational characteristics, computed from these fractions and other input data. For instance, we denote by $\Delta(\mathbf{D}, \mathbf{t})$ the *detection rate* of $(\mathbf{D}, \mathbf{t})$, which is the expected fraction of "bad" containers checked manually, and by $\Phi(\mathbf{D}, \mathbf{t})$ the *rate of false positives*, which is the expected fraction of "good" containers checked manually (unnecessarily). We can also compute the expected *per-container-inspection cost* $C(\mathbf{D}, \mathbf{t})$, the expected *per-container-inspection time* $T(\mathbf{D}, \mathbf{t})$, and expected *sensor loads* $U_s(\mathbf{D}, \mathbf{t})$ for $s \in \mathcal{S}$. Let us remark that we assume the proportion of bad containers to be negligible. Consequently, we can base, without any serious loss of generality, the computation of these

operational characteristics only on the flow of "good" containers (with the exception of the detection rate, which is computed solely from the "bad" container flow).

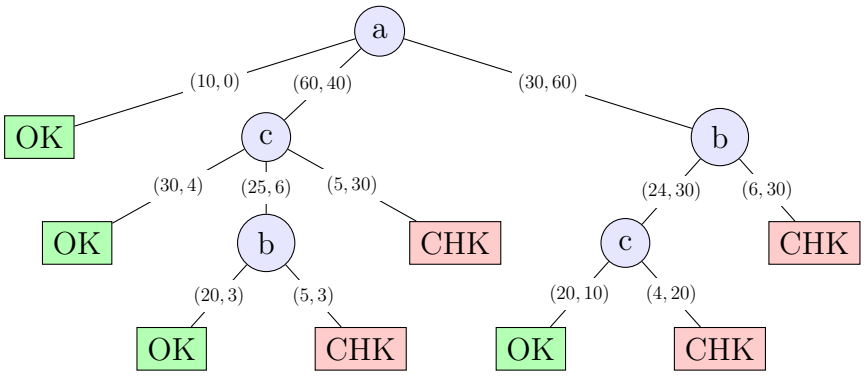


Figure 3: A pure inspection strategy utilizing three sensors $\mathcal{S} = \{a, b, c\}$, and a particular threshold selection. Along each edge we indicate the percentages of "good" and "bad" containers that receive the corresponding readings.

These notions will be defined precisely, later in the paper. Here we illustrate them by using a numerical example. Assume that we choose the thresholds $\mathbf{t}$ for the decision tree $\mathbf{D}$ shown in Figure 2, such that the percentages of "good" and "bad" containers receiving particular readings are as shown in Figure 3. Then we get the following values:

$$\begin{aligned}
\Delta(\mathbf{D}, \mathbf{t}) &= 0.30 + 0.03 + 0.20 + 0.30 = 0.83 \\
\Phi(\mathbf{D}, \mathbf{t}) &= 0.05 + 0.05 + 0.04 + 0.06 = 0.20 \\
U_a(\mathbf{D}, \mathbf{t}) &= 1.00 \\
U_b(\mathbf{D}, \mathbf{t}) &= 0.25 + 0.30 = 0.55 \\
U_c(\mathbf{D}, \mathbf{t}) &= 0.60 + 0.24 = 0.84
\end{aligned}$$

Based on these we can compute $C(\mathbf{D}, \mathbf{t})$ and $T(\mathbf{D}, \mathbf{t})$ as follows

$$\begin{aligned}
C(\mathbf{D}, \mathbf{t}) &= \sum_{s \in \mathcal{S}} C_s U_s(\mathbf{D}, \mathbf{t}) + C_{CHK} \Phi(\mathbf{D}, \mathbf{t}) \\
&= C_a + 0.55 C_b + 0.84 C_c + 0.20 C_{CHK}
\end{aligned}$$

$$\begin{aligned}
T(\mathbf{D}, \mathbf{t}) &= \sum_{s \in \mathcal{S}} T_s U_s(\mathbf{D}, \mathbf{t}) + T_{CHK} \Phi(\mathbf{D}, \mathbf{t}) \\
&= T_a + 0.55 T_b + 0.84 T_c + 0.20 T_{CHK}
\end{aligned}$$

where $C_s$ and $T_s$ respectively are the average inspection cost and inspection time of a single container on sensor $s \in \mathcal{S}$, and $C_{CHK}$ and $T_{CHK}$ respectively are the cost and time of inspecting manually a container.

## 2.3 Finding the Best Pure Inspection Strategy

Using the notation introduced above, we can formulate the problem of finding a best pure inspection strategy as follows:

$$\begin{aligned} \min C(\mathbf{D},\mathbf{t}) &= \sum_{s\in\mathcal{S}} C_s U_s(\mathbf{D},\mathbf{t}) + C_{CHK}\Phi(\mathbf{D},\mathbf{t}) \\ \textit{subject to} & \\ \Delta(\mathbf{D},\mathbf{t}) &\geq \Delta_0 \end{aligned}$$

where the optimization runs over all possible decision trees $\mathbf{D}$, and all possible threshold selections $\mathbf{t}$ for $\mathbf{D}$.

This model was considered by [4] in the simplifying case of binary sensor outputs, assuming that 1) if a particular combination of container observations is selected by for CHK, then any combination of observations in which each observation is "equally or more suspicious" would also be selected for CHK, and 2) whether to CHK or OK depends on all of the sensors. For the case of four sensors observing four independent attributes, there turn out to be 114 feasible selection expressions. These can be implemented by 11,808 distinct decision trees. For each of these trees a nonlinear optimization problem was solved to obtain the best threshold selection by a grid-search method, and the best decision tree was identified.

A test case was created assuming standard normal distribution of readings for bad containers, and normal distribution $N(K_s, 1)$ of readings for good containers at sensors $s = 1, 2, 3, 4$, where the first sensor can, however, recognize only half of bad containers. Discriminating power (the number of standard deviations separating good and bad containers on that sensor) and cost factors (notionally in dollars per scan) of each of four sensors represent generic sensor types. Sensor 1 has good discriminating power (for those containers that it can recognize at all) and is inexpensive, sensor 2 has low discriminating power but is inexpensive, sensor 3 has medium discriminating power and intermediate cost, and sensor 4 is powerful and expensive.

The test case parameters are as follows:

$$\begin{array}{rclrcl} K_1 &=& 4.37, & C_1 &=& 0.32 \\ K_2 &=& 1.53, & C_2 &=& 0.92 \\ K_3 &=& 2.9, & C_3 &=& 57 \\ K_4 &=& 4.6, & C_4 &=& 176 \end{array}$$

with $C_{CHK} = 600$ and $\Delta_0 = .815$.

For each of the 11,806 feasible BDT's, a solver is used to find the set of four threshold settings $\{t_1, t_2, t_3, t_4\}$ that minimize the combined cost of observations and false positives for a specified minimum detection probability.

Ten of the best thirteen BDT's implement the selection logic that suggests a manual inspection of those containers for which either A) sensor 2, sensor 3, and sensor 4 read high; or B) sensor 2, sensor 3, and sensor 1 read high; or C) sensor 1 and sensor 4 read high. There are 148 distinct feasible BDT's that implement this selection logic. The best 7 give

a cost of $13.56 per container and the following optimized threshold settings and utilization associated with each sensor:

$$\begin{array}{rclcrcl} t_1 & = & 2.6, & & U_1 & = & 1 \\ t_2 & = & 1.07, & & U_2 & = & 1 \\ t_3 & = & 1.16, & & U_3 & = & 0.14 \\ t_4 & = & 2.2, & & U_4 & = & 0.02 \end{array}$$

Figure 4 describes the optimal pure strategy, where containers with the reading lower than a threshold follow the left branch of a node. The remaining six out of seven best BDTs can be found in the Appendix (see Figures 15-20).

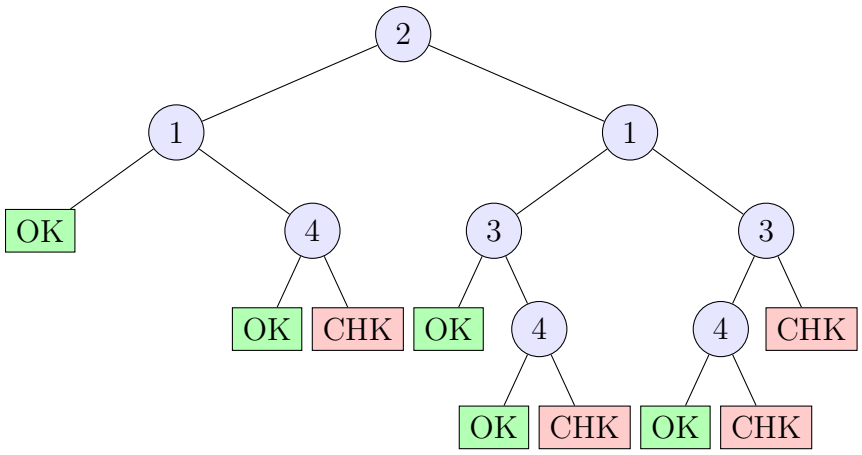


Figure 4: A decision tree representing optimal pure strategy for four sensors with one threshold each.

The worst 14 BDT configurations implementing this logic are so poor that they give a cost per container in excess of $148 (these poor BDT's place sensor 3 or 4 at node 1). Many seemingly quite different BDT's were found to implement near optimal fitness, although at very differently-tuned threshold levels.

From the results of [4] we can conclude also that the above model has several limitations. First of all, it is computationally very expensive, and it is doubtful that one could solve it for $n \geq 5$ sensors. Secondly, it might be substantially better to allow more than one thresholds at each of the sensors, which makes the formulation even less tractable computationally. Last, but not least, using a pure strategy for container inspection may not be the best thing to do. We will argue in the subsequent sections that the use of mixed strategies (combinations of several pure strategies) not only can improve on detection rates, without increasing costs, times, etc., but is also less vulnerable to adaptation of packaging, etc.

## 2.4 The Advantage of Mixed Inspection Strategies

As we remarked earlier, pure inspection strategies do not necessarily provide us with the best possible solutions, that is with the most effective utilization of our resources. We illustrate in Figure 5 the mixing principle with the simplest possible case: a single sensor. We choose

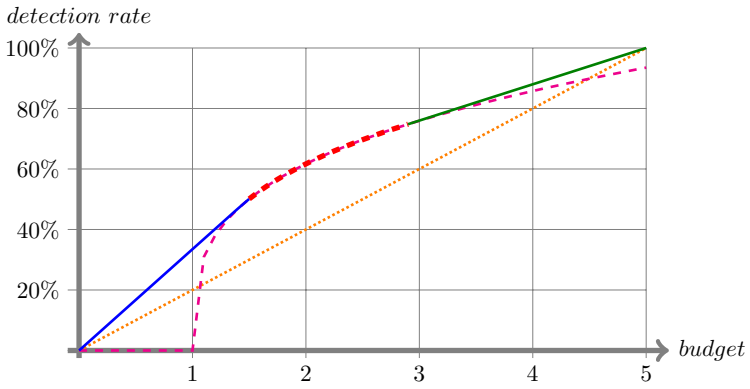


Figure 5: Mixing pure strategies. Random manual checks are plotted as a dotted line, while the single sensor based best strategy is drawn by a dashed curve. The best mixed strategy has three segments. The first one, shown by a solid line, is a mixture of the single sensor strategy with the strategy letting every container go without any checking. The next segment, shown as a bold dashed line, coincides with the single sensor strategy, while the last segment, shown again as a solid line, is a mixture of the single sensor strategy with the one in which we manually check all containers.

a relatively strong sensor for which the distributions are:

$$f_b(y) = 3e^{-3y} \quad \text{and} \quad f_g(y) = 10e^{-10y}.$$

The ratio $10/3e^{-7y}$ corresponds to a separation of 3.5 standard deviations.

The three pure strategies are: unpack everything manually (until the budget is exhausted; denoted by CHK); use the sensor on all the containers and with the remaining budget, unpack them manually in decreasing order of the odds that they are bad (that is, in decreasing order of the sensor reading $y$); let the containers go without any checking (denoted by OK). To make a more readable diagram we set the cost (per container) of using the sensor to be 1, and the cost of unpacking to be 5. The performance of the first two pure strategies is shown in figure 5, while OK clearly has a detection rate equal to zero. The best mixed strategy is also shown in the figure. At very low budgets it is a mixed use of the compound strategy and OK (no examination at all). This can be represented by a diagram as the first decision tree in Figure 6, where the small square represents the (random) mixing. As the budget increases we encounter a short range of budgets where the optimal solution tracks the compound strategy represented by the second decision tree in Figure 6. Finally the mixed strategy becomes a

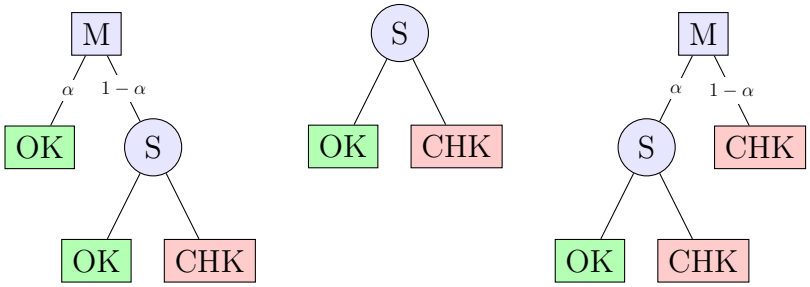


Figure 6: Three mixed strategies corresponding to Figure 5.

mixture of CHK (unpacking manually) and the compound strategy, represented by the third decision tree in Figure 6.

The same structure applies to any case in which there is a single constraint. In general however there will be several curves contributing to the determination of the convex hull. In terms of the polyhedron, the set of vertices determining the optimal point changes from time to time as the budget increases. Note however that as the budget increases, one pure strategy drops out of the optimal mixture, while the other remains active and then begins to mix with a new pure strategy.

# 3 A Linear Programming Model for Finding the Best Mixed Inspection Strategy

Following the idea of fixed-grid search of [4], let us assume that for each sensor $s \in \mathcal{S}$ we fix a partition

$$\mathbf{P}_s \;=\; \{X_j^s | j = 1, ..., k(s)\}$$

of the set of reals, and retain only the index $j$ for each container such that $R_s(C) \in X_j^s$, and not the actual sensor reading $R_s(C)$. Let us then denote by $\mathcal{D}$ the set of all possible decision trees utilizing the given partitions. For instance, Figure 7 shows a possible decision tree for the case when $\mathcal{S} = \{a, b, c\}$ and $|\mathbf{P}_a| = 3$, $|\mathbf{P}_b| = 2$ and $|\mathbf{P}_c| = 4$. Note that some of the ranges could be merged in a decision tree $\mathbf{D} \in \mathcal{D}$. Note also that $\mathbf{D}$ now incorporates the information about the threshold selection, hence each $\mathbf{D} \in \mathcal{D}$ represents an inspection strategy, which we denoted earlier by $(\mathbf{D}, \mathbf{t})$.

Fixing the partitions of the sensor readings incorporated the grid-search over the grid corresponding to the partitions $\mathbf{P}_s$, $s \in \mathcal{S}$ into our notations. This makes it possible to formulate the problem of finding a best mixed inspection strategy as a very large scale linear programming problem.

Let us denote by $x_{\mathbf{D}}$ the fractions of containers which we plan to inspect by inspection strategy $\mathbf{D}$, for $\mathbf{D} \in \mathcal{D}$. Clearly, we have $\sum_{\mathbf{D}\in\mathcal{D}} x_{\mathbf{D}} \;=\; 1$ and $x_{\mathbf{D}} \geq 0$ for all $\mathbf{D} \in \mathcal{D}$. Ideally,

if we would like to stick to a single decision tree as our inspection strategy, we would like to have only one of these $x_{\mathbf{D}}$ variables to be positive. It turns out however, as we noted in the previous section, that it is advantageous to consider *mixed strategies*, i.e., non-integral values for the variables.

If $\mathbf{x} = (x_{\mathbf{D}}|\mathbf{D} \in \mathcal{D})$ represents a mixed strategy, then we have

$$\Delta(\mathbf{x}) = \sum_{\mathbf{D}\in\mathcal{D}} \Delta(\mathbf{D})x_{\mathbf{D}}$$

as its corresponding detection rate,

$$C(\mathbf{x}) = \sum_{\mathbf{D}\in\mathcal{D}} C(\mathbf{D})x_{\mathbf{D}}$$

as its corresponding average per-container inspection cost,

$$T(\mathbf{x}) = \sum_{\mathbf{D}\in\mathcal{D}} T(\mathbf{D})x_{\mathbf{D}}$$

as its corresponding average per-container inspection time,

$$\Phi(\mathbf{x}) = \sum_{\mathbf{D}\in\mathcal{D}} \Phi(\mathbf{D})x_{\mathbf{D}}$$

as its corresponding average rate of false positives, and

$$U_s(\mathbf{x}) = \sum_{\mathbf{D}\in\mathcal{D}} U_s(\mathbf{D})x_{\mathbf{D}}$$

as its corresponding average load of sensor $s$, for $s \in \mathcal{S}$.

It is important to note that all these measures are linear functions of the vector $\mathbf{x}$. Consequently, we can write (in terms of $\mathbf{x}$) the problem of finding a (mixed) inspection policy with the highest detection rate, subject to e.g., budget and capacity constraints as a linear programming problem:

$$\begin{aligned}
&\max \sum_{\mathbf{D}\in\mathcal{D}} \Delta(\mathbf{D})x_{\mathbf{D}} \\
&\textit{subject to} \\
&\qquad \sum_{\mathbf{D}\in\mathcal{D}} C(\mathbf{D})x_{\mathbf{D}} \leq B \\
&\qquad \sum_{\mathbf{D}\in\mathcal{D}} U_s(\mathbf{D})x_{\mathbf{D}} \leq K(s) \quad \text{for all} \quad s \in \mathcal{S} \\
&\qquad\qquad \sum_{\mathbf{D}\in\mathcal{D}} x_{\mathbf{D}} = 1 \\
&\qquad\qquad\qquad x_{\mathbf{D}} \geq 0 \qquad \text{for all } \mathbf{D} \in \mathcal{D}.
\end{aligned} \tag{2}$$

Of course, we could also include constraints on the per-container inspection time, or on the rate of false positives, etc.

Similarly, the problem of finding the inspection strategy which minimizes the average per-container inspection costs, while still providing a prescribed high detection rate of $1-\epsilon$ and not exceeding the available load capacities for each sensor can also be written as a linear programming problem:

$$\begin{aligned}
&\min \sum_{\mathbf{D}\in\mathcal{D}} C(\mathbf{D})x_{\mathbf{D}} \\
&\textit{subject to} \\
&\begin{aligned}
\sum_{\mathbf{D}\in\mathcal{D}} \Delta(\mathbf{D})x_{\mathbf{D}} &\geq 1-\epsilon \\
\sum_{\mathbf{D}\in\mathcal{D}} U_s(\mathbf{D})x_{\mathbf{D}} &\leq K(s) \quad \text{for all} \quad s\in\mathcal{S} \\
\sum_{\mathbf{D}\in\mathcal{D}} x_{\mathbf{D}} &= 1 \\
x_{\mathbf{D}} &\geq 0 \qquad \text{for all } \mathbf{D}\in\mathcal{D}.
\end{aligned}
\end{aligned} \tag{3}$$

Many other variants of these problems, involving limits on rates of false positives, inspection times, etc., can be formulated analogously as linear programming problems.

The main practical difficulty in using the above models is their size. The number of variables in these formulations is $|\mathcal{D}| \gg 2^{\prod_{s\in\mathcal{S}} k(s)} \gg 2^{2^{|\mathcal{S}|}}$ (double exponential in the number of sensors). As we recalled earlier, for 4 sensors, and binary partitions [4] enumerated over 11000 decision trees. With binary partitions of the sensor readings however we cannot achieve the best performance. To be able to utilize the power of the above models, we want a much finer partition of sensor readings, so that the optimization will be able to select the best decision sequence for the decision trees appearing in the optimal mixed strategy. If we allow e.g., $k(s)=10$ for all sensors $s\in\mathcal{S}$, the number of variables in (2) (or (3)) will jump to well above 100 million, making it almost impossible to generate all those and write down in full detail the linear programming problems (2) and (3).

On the positive side, there are only $2+|\mathcal{S}|$ constraints in the above formulations, and thus the optimal basic solution $\mathbf{x}^*$ involves at most $2+|\mathcal{S}|$ decision trees with nonzero $x^*_{\mathbf{D}}$ values. This suggests that such a mixed strategy is far from being impractical, since it is not a complicated matter to switch between a small number of decision trees.

The small size of a basic solution also implies that *column generation* might be a good way of solving these very large linear programming problems. This technique was introduced first by Gilmore and Gomory [5, 6] for the solution of cutting stock problems, but since then column generation has become a standard approach, used widely and efficiently in numerous applications. For the sake of completeness we describe briefly the main ideas.

Let us consider e.g., problem (2) and its linear programming dual, which has only $2+|\mathcal{S}|$ variables, $\alpha$ corresponding to the budget constraint, $\beta_s$, $s \in \mathcal{S}$ corresponding to the load constraints, and $\gamma$ corresponding to the last balance equality. Using these we can write the dual of (2) as follows:

$$\begin{aligned} &\min \; B\,\alpha \; + \; \sum_{s\in\mathcal{S}} K(s)\,\beta_s \; + \; \gamma \\ &\textit{subject to} \\ &\quad C_{\mathbf{D}}\,\alpha \; + \; \sum_{s\in\mathcal{S}} U_s(\mathbf{D})\,\beta_s \; + \; \gamma \;\; \geq \;\; \Delta(\mathbf{D}) \qquad \text{for all } \mathbf{D}\in\mathcal{D} \\ &\qquad\qquad \alpha,\gamma,\beta_s \;\; \geq \;\; 0 \qquad\qquad \text{for all } s\in\mathcal{S}. \end{aligned} \tag{4}$$

Let us now consider a small subset $\widehat{\mathcal{D}} \subseteq \mathcal{D}$ of all decision trees, and suppose that we have solved (2) restricted to this set (i.e., we replace $\mathcal{D}$ everywhere in (2) by $\widehat{\mathcal{D}}$). Let us denote by $\mathbf{x}^*$ the optimal solution to this restricted variant of (2), and let $\alpha^*$, $\beta_s^*$, $s \in \mathcal{S}$, and $\gamma^*$ denote the corresponding optimal solution of the dual of the restricted problem (i.e., where we keep constraints in (4) only for $\mathbf{D} \in \widehat{\mathcal{D}}$).

The theory of linear programming then implies (see e.g., [3]) that $\mathbf{x}^*$ is an optimal solution for the unrestricted problem (2) if and only if

$$\min_{\mathbf{D}\in\mathcal{D}} \left[ C_{\mathbf{D}}\,\alpha^* \; + \; \sum_{s\in\mathcal{S}} U_s(\mathbf{D})\,\beta_s^* \; + \; \gamma^* \; - \; \Delta(\mathbf{D}) \right] \;\geq\; 0. \tag{5}$$

If this is the case, we can stop, and $\mathbf{x}^*$ is our optimal solution. Otherwise, we choose a decision tree $\mathbf{D}^*$ for which the minimum in (5) is attained, increment $\widehat{\mathcal{D}} = \widehat{\mathcal{D}} \cup \{\mathbf{D}^*\}$, and return to the solution of the restricted (2) problem. Experience shows that on average, this approach generates only a small number of the variables in (2). In fact, as a consequence of the results of Khachiyan [8], the number of times we need to solve (5) is limited by a polynomial of the rank of (2), i.e., a polynomial function of $|\mathcal{S}|$ in our case.

However, for this to work, one has to be able to solve subproblem (5) efficiently. In its current form problem (5) still seem to require the complete enumeration of all decision trees, which is an approach that does scale well, as we noted before. In the following sections we show that problem (5) can in fact be reformulated such that it can be solved efficiently.

# 4 Histories and Decision Trees

Let us note that each container $C$, subjected to a particular inspection policy $\mathbf{D}$ will produce a *history* $\pi(C,\mathbf{D})$ of sequential sensor readings, and a final decision. We shall represent such a history as an ordered sequence of pairs $(s,j)$, where $s$ is a sensor on which $C$ was tested and

$j$ is the retained index such that $R_s(C) \in X_j^s$. The order of these pairs in the history $\pi(C, \mathbf{D})$ is the order in which $C$ was tested with the different sensors. Finally, the last element of $\pi(C, \mathbf{D})$ is the final decision made on $C$, i.e, either "OK" if $C$ was let go, or "CHK" if $C$ was manually checked.

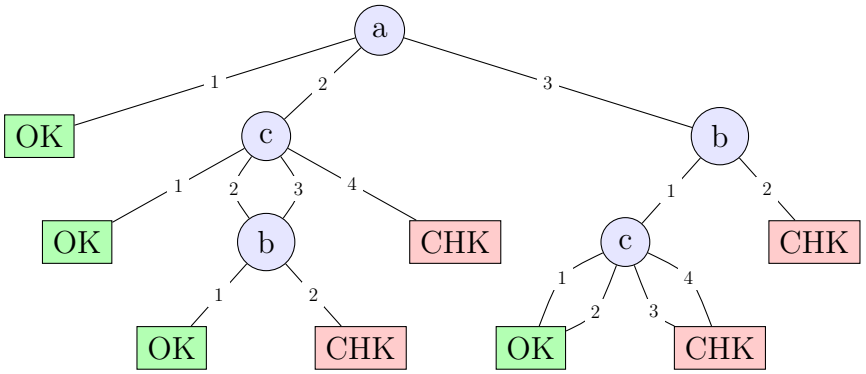


Figure 7: A decision tree utilizing three sensors $\mathcal{S} = \{a, b, c\}$, and partitions of the sensor readings such that $|\mathbf{P}_a| = 3$, $|\mathbf{P}_b| = 2$ and $|\mathbf{P}_c| = 4$.

Let us consider the decision tree of Figure 7, using sensors $\mathcal{S} = \{a, b, c\}$ and partitions of sensor readings such that $|\mathbf{P}_a| = 3$, $|\mathbf{P}_b| = 2$ and $|\mathbf{P}_c| = 4$. We can see that there are 12 different possible histories, which we list here below:

$$\begin{aligned}
\pi_1 &= \{(a,1), OK\} \\
\pi_2 &= \{(a,2), (c,1), OK\} \\
\pi_3 &= \{(a,2), (c,2), (b,1), OK\} \\
\pi_4 &= \{(a,2), (c,2), (b,2), CHK\} \\
\pi_5 &= \{(a,2), (c,3), (b,1), OK\} \\
\pi_6 &= \{(a,2), (c,3), (b,2), CHK\} \\
\pi_7 &= \{(a,2), (c,4), CHK\} \\
\pi_8 &= \{(a,3), (b,1), (c,1), OK\} \\
\pi_9 &= \{(a,3), (b,1), (c,2), OK\} \\
\pi_{10} &= \{(a,3), (b,1), (c,3), CHK\} \\
\pi_{11} &= \{(a,3), (b,1), (c,4), CHK\} \\
\pi_{12} &= \{(a,3), (b,2), CHK\}
\end{aligned}$$

For a given decision tree $\mathbf{D}$ let us denote by $\Pi(\mathbf{D})$ the (finite) set of possible container histories, and let $\Pi$ denote the set of all those histories, which may appear in some decision trees. Note that our notation assumes that we have fixed the set $\mathcal{S}$ of sensors and also the partition $\mathbf{P}_s$ of sensor readings, for each $s \in \mathcal{S}$.

To each history $\pi \in \Pi(\mathbf{D})$ let us associate two parameters:

$$g_\pi \;=\; Prob\left( \bigwedge_{(s,j)\in\pi} \left(R_s(C) \in X_j^s\right) \;\middle|\; C \text{ is of type "}\textit{good}\text{"} \right)$$

denoting the average fraction of "good" containers that have history $\pi$, and

$$b_\pi \;=\; Prob\left( \bigwedge_{(s,j)\in\pi} \left(R_s(C) \in X_j^s\right) \;\middle|\; C \text{ is of type "}\textit{bad}\text{"} \right)$$

denoting the average fraction of "bad" containers with this history. Since every container of either type must have some specific path, these parameters satisfy the equalities

$$\sum_{\pi\in\Pi(\mathbf{D})} g_\pi \;=\; \sum_{\pi\in\Pi(\mathbf{D})} b_\pi \;=\; 1. \tag{6}$$

For instance, if we assume that the sensors are stochastically independent we can compute these parameters from the sensor distributions as follows:

$$g_\pi = \prod_{(s,j)\in\pi} \xi(X_j^s, s, \text{"}\textit{good}\text{"}) \quad \text{and} \quad b_\pi = \prod_{(s,j)\in\pi} \xi(X_j^s, s, \text{"}\textit{bad}\text{"}). \tag{7}$$

Let us add that the stochastic independence of the sensors is not an important assumption. What is needed for our model as input are the quantities $g_\pi$ and $b_\pi$ for all paths $\pi \in \Pi$.

Let us observe next that the performance characteristics of an inspection strategy $\mathbf{D}$ introduced in the previous section are in fact linear functions of the parameters $g_\pi$ and $b_\pi$, $\pi \in \Pi(\mathbf{D})$.

For instance, the detection rate $\Delta(\mathbf{D})$ of an inspection strategy $\mathbf{D}$, that is the probability that a "bad" container will get manually inspected by $\mathbf{D}$, can be written as

$$\Delta(\mathbf{D}) \;=\; \sum_{\substack{\pi\in\Pi(\mathbf{D}) \\ \text{CHK}\in\pi}} b_\pi.$$

Let us recall that, we assume that truly dangerous containers are very rare. Consequently, all other operating characteristics of an inspection strategy $\mathbf{D}$ will be dominated by the "good" containers (which are the vast majority of containers actually tested by this policy).

The rate of false positives $\Phi(\mathbf{D})$ of $\mathbf{D}$, that is the expected fraction of "good" containers that end up manually inspected (unnecessarily) can be written as

$$\Phi(\mathbf{D}) \;=\; \sum_{\substack{\pi\in\Pi(\mathbf{D}) \\ \text{CHK}\in\pi}} g_\pi.$$

For a sensor $s \in \mathcal{S}$ the sensor load $U_s(\mathbf{D})$ is the fraction of containers that are actually tested on sensor $s$ by the inspection policy $\mathbf{D}$, and it can be written as

$$U_s(\mathbf{D}) \;=\; \sum_{\substack{\pi \in \Pi(\mathbf{D}) \\ (s,j) \in \pi \text{ for some } j}} g_\pi .$$

The average per-container inspection cost $C(\mathbf{D})$ can be computed as

$$C(\mathbf{D}) \;=\; \sum_{s \in \mathcal{S}} C_s U_s(\mathbf{D}) \;+\; C_{CHK}\Phi(\mathbf{D}),$$

where, as before, $C_s$ denotes the average per-container inspection cost of sensor $s$, and $C_{CHK}$ denotes the average per-container cost of manual inspection.

Analogously, the average per-container inspection time $T(\mathbf{D})$ can be written as

$$T(\mathbf{D}) \;=\; \sum_{s \in \mathcal{S}} T_s U_s(\mathbf{D}) \;+\; T_{CHK}\Phi(\mathbf{D}),$$

where, as before, $T_s$ denotes the average per-container inspection time of sensor $s$, and $T_{CHK}$ denotes the average per-container time of manual inspection.

There are several further operational characteristics which might be important to consider (such as container delays; separate detection rates for different types of dangerous containers when we do not know the relative frequency of those, separate false positive rates for different types of "good" containers, etc.). Since we do not have at this time reliable data for those, and since we want to keep our presentation simple, we do not consider other measures in this study.

# 5 The Polytope of Decision Trees

Let us consider a given set $\mathcal{S}$ of sensors, a fixed partition $\mathbf{P}_s$ of the possible sensor readings for each $s \in \mathcal{S}$, and recall that we denote by $\Pi$ the set of all possible container histories, while $\Pi(\mathbf{D})$ denotes the container histories of containers inspected by strategy $\mathbf{D}$. In other words, we have

$$\Pi \;=\; \bigcup_{\mathbf{D} \in \mathcal{D}} \Pi(\mathbf{D}).$$

Furthermore, any decision tree $\mathbf{D}$ can, equivalently, be represented by the vectors $(g_\pi \mid \pi \in \Pi(\mathbf{D}))$ and $(b_\pi \mid \pi \in \Pi(\mathbf{D}))$. Let us extend these vectors with zeros, and define in this way $\mathbf{g}(\mathbf{D}), \mathbf{b}(\mathbf{D}) \in \mathbb{R}^{\Pi}$, where

$$\mathbf{g}(\mathbf{D})_\pi \;=\; \begin{cases} g_\pi & \text{if } \pi \in \Pi(\mathbf{D}) \\ 0 & \text{if } \pi \in \Pi \setminus \Pi(\mathbf{D}) \end{cases} \quad \text{and} \quad \mathbf{b}(\mathbf{D})_\pi \;=\; \begin{cases} b_\pi & \text{if } \pi \in \Pi(\mathbf{D}) \\ 0 & \text{if } \pi \in \Pi \setminus \Pi(\mathbf{D}) \end{cases}$$

In fact a mixed strategy $\mathbf{x} = (x_{\mathbf{D}} \mid \mathbf{D} \in \mathcal{D})$ can also be represented naturally by the fractions of "good" and "bad" containers having a particular history $\pi \in \Pi$. The corresponding vectors $\mathbf{g}(\mathbf{x}), \mathbf{b}(\mathbf{x}) \in \mathbb{R}^{\Pi}$ are convex combinations:

$$\mathbf{g}(\mathbf{x}) \;=\; \sum_{\mathbf{D}\in\mathcal{D}} x_{\mathbf{D}}\mathbf{g}(\mathbf{D}) \quad \text{and} \quad \mathbf{b}(\mathbf{x}) \;=\; \sum_{\mathbf{D}\in\mathcal{D}} x_{\mathbf{D}}\mathbf{b}(\mathbf{D})$$

Thus, we can associate naturally two polytopes, $P_g$ and $P_b$, to the set of sensors and the fixed partitions of their readings, corresponding to the two container types we considered. These polytopes are defined as

$$P_g = ConvHull\,\langle \mathbf{g}(\mathbf{D}) \mid \mathbf{D} \in \mathcal{D}\rangle \;=\; \left\{ \mathbf{g}(\mathbf{x}) \;\middle|\; \begin{array}{c} \sum_{\mathbf{D}\in\mathcal{D}} x_{\mathbf{D}} \;=\; 1 \\ x_{\mathbf{D}} \geq 0 \text{ for all } \mathbf{D} \in \mathcal{D} \end{array} \right\}$$

and

$$P_b = ConvHull\,\langle \mathbf{b}(\mathbf{D}) \mid \mathbf{D} \in \mathcal{D}\rangle \;=\; \left\{ \mathbf{b}(\mathbf{x}) \;\middle|\; \begin{array}{c} \sum_{\mathbf{D}\in\mathcal{D}} x_{\mathbf{D}} \;=\; 1 \\ x_{\mathbf{D}} \geq 0 \text{ for all } \mathbf{D} \in \mathcal{D} \end{array} \right\}.$$

To characterize these polytopes in terms of a linear system, we need to introduce more notation and terminology.

Recall that a history $\pi$ of a container $C$ is an ordered sequence of pairs $(s, j)$, where $s$ is a sensor and $j$ is the index of one of the parts of the partition $\mathbf{P}_s = \{X_i^s \mid i = 1, ..., k(s)\}$, and the last element of $\pi$ is one of the symbols "OK" and "CHK".

Let us now consider an arbitrary ordered sequence $\mu$ of pairs $(s, j)$, where $s \in \mathcal{S}$, and $j$ is an index satisfying $1 \leq j \leq k(s)$, and such that no sensor appears twice in $\mu$. Let us call such a sequence a *pre-history*, and for two such sequences $\mu, \nu$ let us write $\mu \prec \nu$ if $\mu$ is an initial subsequence of $\nu$.

For instance, if we have the history $\pi = ((a, 3), (b, 1), (c, 1), OK)$, then the sequences $\mu_1 = ((a, 3))$, $\mu_2 = ((a, 3), (b, 1))$ and $\mu_3 = ((a, 3), (b, 1), (c, 1))$ are all pre-histories (of $\pi$), and we have

$$\mu_1 \prec \mu_2 \prec \mu_3 \prec \pi.$$

Clearly, if $\mu$ and $\nu$ are pre-histories, and they do not involve the same sensors, then their concatenation $(\mu, \nu)$ is also a pre-history.

The following statement provides a linear characterization of the polytopes $P_g$ and $P_b$. This result turns out to be very useful for solving problem (2) (and other variants). We state the characterization only for polytope $P_g$, since the perfectly analogous claim can easily be shown in the same way for $P_b$, as well. To simplify notation, we assume that the sensors are stochastically independent, and parameters $g_\pi$ and $b_\pi$ are computed by (7) for all $\pi \in \Pi$. Let us add however that this is just a technical detail, and analogous linear description for the polytopes $P_g$ and $P_b$ can be derived for stochastically dependent sensors, as well.

**Theorem 1** *Given $\mathbf{y} \in \mathbb{R}^{\Pi}$, we have $\mathbf{y} \in P_g$ if and only if $\mathbf{y}$ satisfies the following equalities:*

$$\sum_{\pi\in\Pi} y_\pi \;=\; 1 \tag{8}$$

*and*

$$\sum_{\pi:(\nu,(s,j))\prec\pi} y_\pi \;=\; \xi(X_j^s, s, \text{“}good\text{”}) \left( \sum_{i=1}^{k(s)} \sum_{\pi:(\nu,(s,i))\prec\pi} y_\pi \right) \tag{9}$$

*for all sensors $s \in \mathcal{S}$, for all indices $1 \leq j \leq k(s)$, and for all pre-histories $\nu$ not involving sensor $s$.*

**Proof**. We prove the statement by “peeling off” vectors $\mathbf{g}(\mathbf{D})$ from $\mathbf{y}$, until we arrive to $\mathbf{y} = 0$, each time decreasing the number of histories $\pi \in \Pi$ for which $y_\pi > 0$.

To be able to do this “peeling off” process, we set a real parameter $\sigma = 1$, and claim that if $\mathbf{y}$ satisfies equalities (8) and (9), then there exists a decision tree $\mathbf{D} \in \mathcal{D}$ such that $y_\pi > 0$ for all $\pi \in \Pi(\mathbf{D})$. For such a decision tree $\mathbf{D}$, we set

$$\lambda = \min_{\pi\in\Pi(\mathbf{D})} \frac{y_\pi}{g_\pi}.$$

Due to (6) and (8), we must have $0 \leq \lambda \leq 1$. Let us set $x_{\mathbf{D}} = \sigma\lambda$. If $\lambda = 1$, then we must have $\mathbf{y} = \mathbf{g}(\mathbf{D})$, and our proof is complete. Since both $\mathbf{y}$ and $\mathbf{g}(\mathbf{D})$ satisfies equalities (8) and (9), by setting

$$\mathbf{y}' = \frac{\mathbf{y} - \lambda\mathbf{g}(\mathbf{D})}{1-\lambda} \quad \text{and} \quad \sigma' = \sigma(1-\lambda).$$

we obtain a vector $\mathbf{y}'$ which also satisfies (8) and (9), and which has at least one less history with a positive $y'_\pi$ value than vector $\mathbf{y}$ had in the previous step. Thus, replacing $\mathbf{y}$ by $\mathbf{y}'$, $\sigma$ by $\sigma'$, and repeating the above steps, after a finite number of iterations we can arrive to a convex combination of the vectors $\mathbf{g}(\mathbf{D})$, $\mathbf{D} \in \mathcal{D}$ which is equal to $\mathbf{y}$, completing the proof of our statement.

What is left is to show our claim, namely that if $\mathbf{y}$ satisfies equalities (8) and (9), then there exists a decision tree $\mathbf{D} \in \mathcal{D}$ such that $\mathbf{y}_\pi > 0$ whenever $g_\pi > 0$ (recall that $\mathbf{g}(\mathbf{D}) = (g_\pi \mid \pi \in \Pi(\mathbf{D}))$.

We prove this claim by induction on the number of sensors. Clearly, if we have only one sensor, then the claim is almost trivial, since we only have to try all possible final decisions at the leafs of the only possible trivial decision tree (consisting of only one node).

Let us now assume that we proved this claim for up to $k$ sensors, and consider the case when $|\mathcal{S}| = k+1$. Consider an arbitrary history $\pi \in \Pi$ for which $y_\pi > 0$, and let $s \in \mathcal{S}$ be the first sensor appearing in this history (i.e., for which $((s,j)) \prec \pi$ for some $1 \leq j \leq k(s)$). Since $\xi(X_i, s, \text{“}good\text{”}) > 0$ for all $i = 1, ..., k(s)$, we must also have histories $\pi' \in \Pi$ with $y_{\pi'} > 0$ and with $((s,i)) \prec \pi'$ for all $i = 1, ..., k(s)$. Let us denote by $\widehat{\pi}$ the history obtained from $\pi$ by deleting the first sensor from $\pi$, and consider the sets

$$\Pi_i \;=\; \{\widehat{\pi} \mid \pi \in \Pi,\ ((s,i)) \prec \pi\}$$

for $i = 1, ..., k(s)$. Each set $\Pi_i$ is the set of all possible histories for the smaller set $\mathcal{S} \setminus \{s\}$ of sensors. Furthermore, for an index $1 \leq i \leq k(s)$ let us define

$$\rho_i = \sum_{\pi\in\Pi,((s,i))\prec\pi} y_\pi$$

and a new vector $\mathbf{y}^i \in \mathbb{R}^{\Pi_i}$ defined by

$$y^i_{\widehat{\pi}} = \frac{1}{\rho_i} y_\pi \quad \text{for all} \quad \widehat{\pi} \in \Pi_i.$$

Then, $\mathbf{y}^i$ is a vector satisfying equations (8) and (9), with respect to sensors $\mathcal{S}\setminus\{s\}$. Thus, we must have a decision tree $\mathbf{D}_i$ such that $\mathbf{y}^i_{\widehat{\pi}} > 0$ for all $\widehat{\pi} \in \Pi(\mathbf{D}_i)$ by our inductive hypothesis. Since this holds for all $i = 1, ..., k(s)$, by gluing these decision trees $\mathbf{D}_i$, $i = 1, ..., k(s)$ to sensor $s$ as a root note, we obtain a decision tree $\mathbf{D}$ rooted at sensor $s$, satisfying our claim. □

# 6 A Computationally Tractable LP Model

To provide an efficient linear programming based scheme, we note that by Theorem 1 every decision tree can be viewed as a particular (vertex) $\mathbf{y} \in P_g$, and conversely, every vector $\mathbf{y} \in P_g$ can be viewed as a mixed strategy. Thus, to solve (5) all we need is to represent $\mathbf{D} \in \mathcal{D}$ by the corresponding $\mathbf{y}$ vector, and perform the minimization over $\mathbf{y} \in P_g$ instead of $\mathbf{D} \in \mathcal{D}$. If we can do this, such that we obtain an expression which is linear in $\mathbf{y}$, then we can solve (5) as a linear programming problem.

To this end, let us associate cost, detection rate, load parameters, etc., to every history $\pi \in \Pi$. Let us denote by $\mathcal{S}(\pi)$ the set of sensors appearing in $\pi$, and define

$$C(\pi) \;=\; \begin{cases} \displaystyle\sum_{s\in\mathcal{S}(\pi)} C_s & \text{if } \text{“}OK\text{”} \in \pi, \\ \\ \displaystyle C_{CHK} + \sum_{s\in\mathcal{S}(\pi)} C_s & \text{if } \text{“}CHK\text{”} \in \pi, \end{cases}$$

$$\Delta(\pi) \;=\; \begin{cases} 0 & \text{if } \text{“}OK\text{”} \in \pi, \\ \\ b_\pi & \text{if } \text{“}CHK\text{”} \in \pi, \end{cases}$$

where $b_\pi$ is defined in (6), and

$$U_s(\pi) \;=\; \begin{cases} 1 & \text{if } s \in \mathcal{S}(\pi), \\ \\ 0 & \text{if } s \notin \mathcal{S}(\pi), \end{cases}$$

for $s \in \mathcal{S}$. Furthermore, to a vector $\mathbf{y} \in P_g$ we associate

$$\begin{aligned} C(\mathbf{y}) \;&=\; \sum_{\pi\in\Pi} C(\pi) y_\pi, \\ \Delta(\mathbf{y}) \;&=\; \sum_{\pi\in\Pi} \Delta(\pi) \frac{y_\pi}{g_\pi}, \text{ and} \\ U_s(\mathbf{y}) \;&=\; \sum_{\pi\in\Pi} U_s(\pi) y_\pi \quad \text{for } s \in \mathcal{S}. \end{aligned}$$

In particular, if $\mathbf{y} = \mathbf{g}(\mathbf{D})$ for a decision tree $\mathbf{D} \in \mathcal{D}$, then we have $C(\mathbf{y}) = C(\mathbf{D})$, $\Delta(\mathbf{y}) = \Delta(\mathbf{D})$ and $U_s(\mathbf{y}) = U_s(\mathbf{D})$ for $s \in \mathcal{S}$, exactly as those parameters were defined in Section 2.2.

Thus, we can rewrite the optimization problem in (5) as the following linear programming problem:

$$\min_{\mathbf{y} \in P_g} \left[ C_{\mathbf{y}}\, \alpha^* \; + \; \sum_{s \in \mathcal{S}} U_s(\mathbf{y})\, \beta_s^* \; + \; \gamma^* \; - \; \Delta(\mathbf{y}) \right]. \tag{11}$$

Note that in the subsequent iterations only the objective function in (11) changes, and hence we can keep the previously optimal basis, and just re-optimize it with the new objective, which perhaps will save time, and increases efficiency.

# 7 Computational Experiments

We solve model A (problem (4), maximization of detection rate subject to given budget) and model B (problem (5), minimization of operating costs per container subject to given detection rate) for 4 sensors and for up to 7 thresholds, assuming unlimited capacities of each sensor.

Using the same data as in [4] we assume that at each sensor bad and good containers have readings distributed normally with variance 1. Readings of good containers have mean 0 for all sensors.

For the first sensor, half of the readings of bad containers have mean $K_1 = 4.37$, another half has the same distribution as good containers (see Figure 8), or in other words sensor 1 can distinguish only one out of two different types of bad containers.

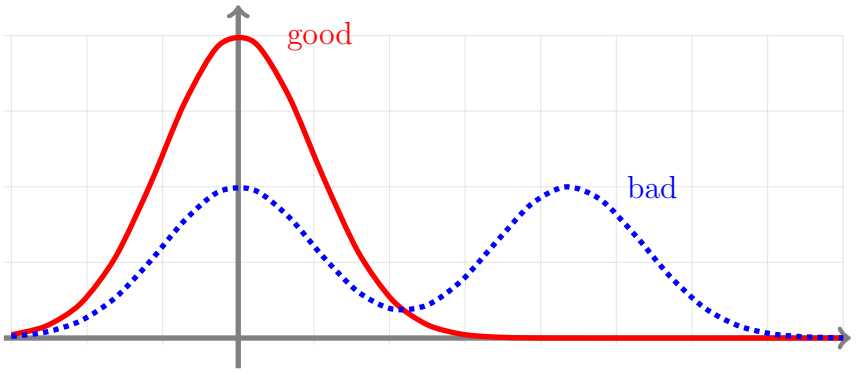


Figure 8: Distributions of sensor readings of sensor 1 for "good" and "bad" sensors, indicated respectively by solid and dotted curves. The cost of inspection of a container by this sensor is $C_1 = 0.32$.

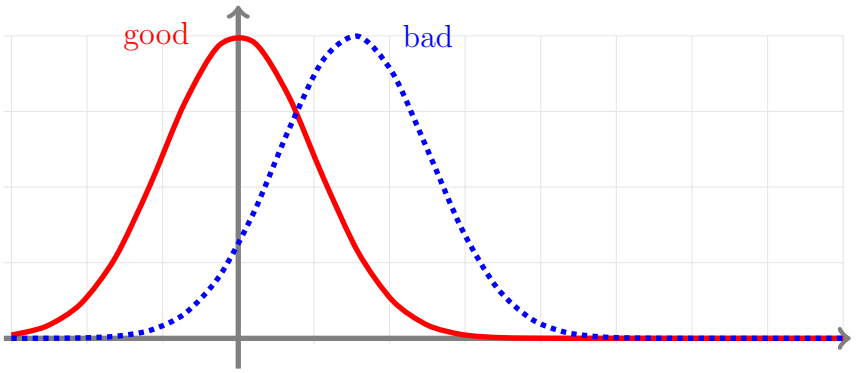


Figure 9: Distributions of sensor readings of sensor 2 for "good" and "bad" sensors, indicated respectively by solid and dotted curves. The cost of inspection of a container by this sensor is $C_2 = 0.92$.

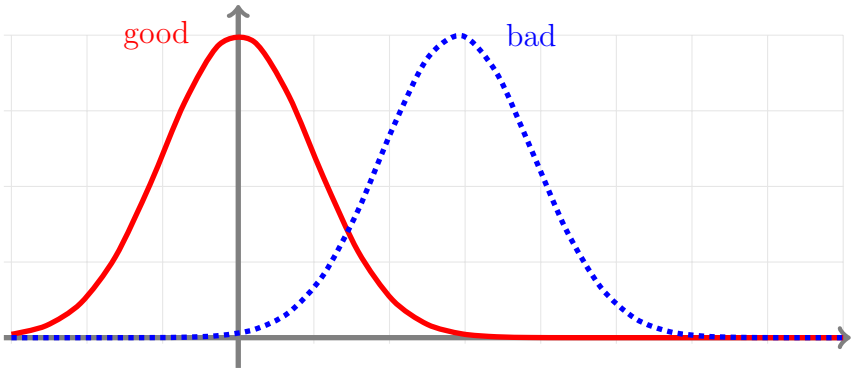


Figure 10: Distributions of sensor readings of sensor 3 for "good" and "bad" sensors, indicated respectively by solid and dotted curves. The cost of inspection of a container by this sensor is $C_3 = 57$.

For bad containers the mean at sensor $s = 2,\ 3,\ 4$ is $K_s$ (see Figures 9,10 and 11), where $K_2 = 1.53$, $K_3 = 2.9$ and $K_4 = 4.6$.

The average per-container inspection costs on the four sensors respectively are $C = (0.32,\ 0.92, 57, 176)$, and $C_{CHK} = \$600$.

Let us recall that since the proportion of "bad" containers is smaller than $10^{-6}$, inspection costs related to "bad" containers are negligible when compared to those of "good" containers. In our model we did not include any inspection cost related to "bad" containers.

Solving model B for detection rate 81.5% and for two-part partitions, where the one threshold for each sensor is chosen at the intersection of the normal distributions for good and bad containers, we obtain that the optimal expected inspection cost per container is

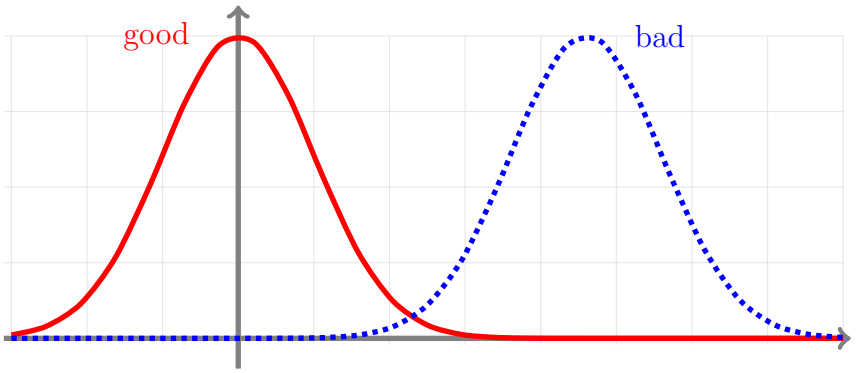


Figure 11: Distributions of sensor readings of sensor 4 for "good" and "bad" sensors, indicated respectively by solid and dotted curves. The cost of inspection of a container by this sensor is $C_4 = 176$.

\$16.75. An optimal decision tree $\mathbf{D}^*$ is shown in Figure 12. As one can see, it is a mixed strategy, or in other words it is a combination of two decision trees, say $\mathbf{D}_1$ and $\mathbf{D}_2$, where 7.9% of the time we first proceed to sensor 1, and 92.1% of the time we first proceed to sensor 2.

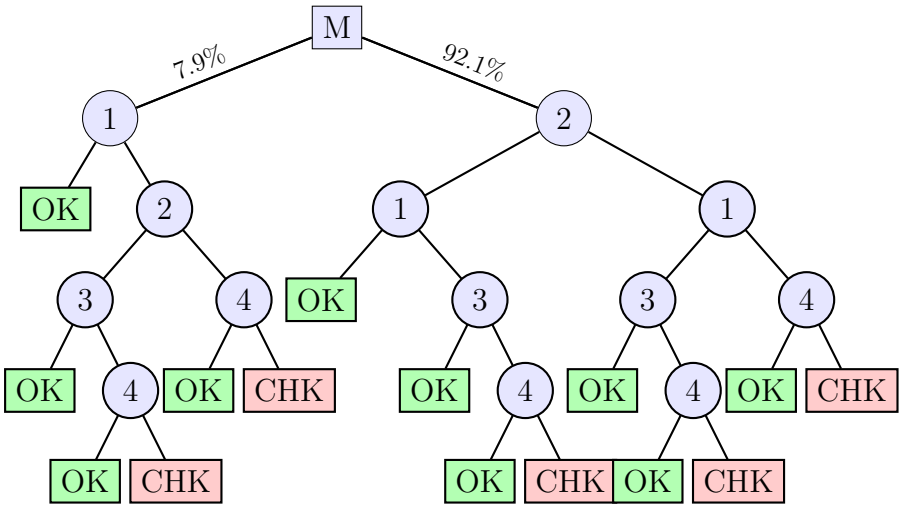


Figure 12: Optimal decision tree $\mathbf{D}^*$ with one threshold, that is a binary partition for each sensor. It is a mixed strategy, since 92.1% of the containers are inspected by a different subtree than the other 7.9%.

Let us note that by allowing up to 7 thresholds for each sensors, the average inspection cost per container drops to 12.07 (see Figure 13), while still guaranteeing a detection rate of

81.5%. A hefty savings of several millions of dollars, if implemented at all US ports. Let us add that this savings can be achieved without any additional investments, and without the deployment of costly new technologies. This assumes only the reorganization of inspection procedures, using the currently used sensor technologies!

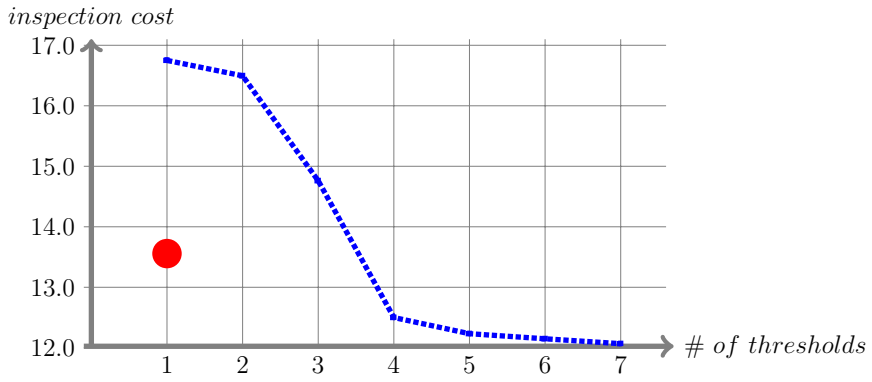


Figure 13: Unit inspection costs as a function of the number of thresholds (same for all sensors), when the detection rate is required to be at least 81.5%. The circle dot indicates the performance of the best threshold-optimized pure strategy found by Stroud and Saeger (2003). It appears that with 7 thresholds improvement approaches a limit which is about 12, a savings of 10% of the cost, through better utilization of all sensor information.

Obviously, achieving higher detection rate requires larger budget. This monotone relation between lower bound on detection rate and optimal operating budget per container is depicted on Figure 14.

Let us close with listing the sizes of the LP problems, and corresponding computing times. We have used the XPRESS-MP package formulating our model in the Mosel modeling language, and solving the large scale linear programming problems by the Newton-barrier method. We did our experiments on a Dell Optiplex 270, with a 3GHz Pentium processor and 2Gb memory.

Let us add that the number of decision trees with a binary partition is over $10^5$; with a partition into 4-4 ranges it is over $10^{15}$; while with 7 ranges at each of the 4 sensors it is well over $10^{50}$. Clearly, enumerative approaches would be impossible at these sizes.

# 8 Conclusions

The results reported herein show several important features of the sensor scheduling problem. We have shown that an approach which makes more detailed use of the specific readings

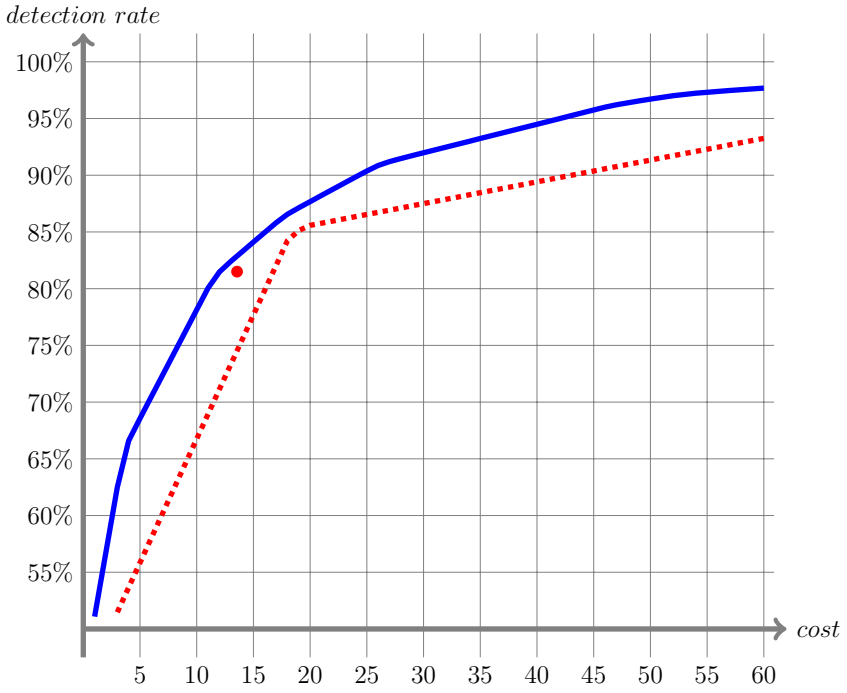


Figure 14: Unit inspection cost – detection rate curves for 1 (dotted line), and 7 (solid line) thresholds, that is when we partition sensor readings into 2, and 8 parts, respectively, for each of the sensors. The circle dot indicates the performance of the threshold optimized solution found by Stroud and Saeger (2003). As we can see in this figure, as well as in figure 13, even with a non-optimized grid selection we can achieve better performance than the best pure strategy if we allow a larger number of thresholds.

on earlier sensor (quantized here by the partition all of containers which have been examined) produces better performance, which increases steadily as the accuracy of the retained information increases. We have also shown that reformulating the problem in terms of a polyhedron in a very high dimensional space, whose points represent combinations of possible paths of a container through the sensor network, leads to a large scale linear programming problem that can be solved efficiently.

Although we have solved the problem here only for particular choices of the sensor performance, and considering only the budgetary constraint, the method extends without difficulty to constraints on the capacity of the individual sensors as well.

Of particular importance is the fact that the use of mixed strategies gives much better performance than is achieved by any pure strategy, particularly in the important region of inadequate budgets. Using mixed strategies means, in practice, randomly assigning containers to one or another path through the collection of sensors. This has some added benefits in real operational situations, as the adversary cannot be assured that any particular container

| Number of | | | | | CPU time | |
|---|---|---|---|---|---|---|
| ranges per sensor | constraints | variables | nonzeros | iterations | setup (sec) | solver (sec) |
| 2 | 318 | 1264 | 6960 | 17 | 0.05 | 0.06 |
| 3 | 1810 | 5424 | 50640 | 25 | 0.20 | 0.58 |
| 4 | 5918 | 15776 | 209056 | 32 | 3.66 | 3.58 |
| 5 | 14658 | 36640 | 630240 | 36 | 30.81 | 23.59 |
| 6 | 30622 | 73489 | 1518794 | 45 | 147.88 | 160.22 |
| 7 | 56978 | 132945 | 3276170 | 53 | 487.38 | 1975.30 |

Table 1: Problem sizes and computational times.

will follow a particular path through the collection of sensors.

# 9 Discussion and Future Research

We have considered here a specific formulation of the overall problem, in which variation in the sensor results is considered to be due to random variation in the containers themselves, and not to variation in the sensors. If there is variation in the sensors, it is sometimes advantageous to apply the same sensor twice, to achieve a more accurate reading. This can clearly be accommodated in our framework, by allowing paths to use the same sensor more than once. It is apparent, from the results reported here, that increasing the resolution of the partition results in better overall performance. The improvement appears to slow with increasing resolution of the partition, and it is presumably asymptotic to the result that would be obtained with infinitely accurate partitions. Further study of the rate of convergence to this limit would be of great value, and calls for some change in approach, some of whose features we can sketch here.

One change is to move from a partition based on defining intervals in terms of the sensor reading to a partition based on the values of the ratio of the two conditional distributions. We expect that this would increase the performance achievable with any given complexity of the partitions. It would also provide a way of dealing with the data from multichannel analyzers, which, in a sense, represent many sensors whose readings are obtained simultaneously.

The method developed here, although the problem exhibits conditional independence of the sensor readings, can be extended to situations in which the readings of the sensors are stochastically dependent, for both the "good" and "bad" containers. At present we have no access to data exhibiting such dependence, and that work will be developed elsewhere.

Our shift from a formulation in which the costs of errors are combined with the costs of operation has, we believe, important practical advantages. The cost of operations are concrete and easily documented. The costs of various types of errors, specifically, false negatives which might permit a dangerous cargo to pass through the inspection site, are matters of debate and conjecture. In situations where these appear as parameters of a model, they are often contested, not on the basis of any concrete information, but rather to bring the results of the model to an acceptable level. In the worst case, such cost parameters are

manipulated to show that whatever budget is currently applied to any monitoring situation is "adequate". By plotting the chance that a dangerous cargo is missed (in fact, is detected) against the cost of operations, we make the most logical separation between the concrete facts and the facts which are subject to differences of opinion. When decision-makers must choose whether to allocate another dollar to inspection, or to some other protective measure, being able to compare both of these in terms of their effect on the chance that something bad will happen provides the most rational possible basis for debate. Although decision-makers do not like to deal in probabilities, every type of serious disturbance, most notably weather disturbances, is in fact stochastic in nature and this is the proper language for analyzing it.

Although this paper has focused on the mathematical aspects of the sensor management problem, we believe that the practical aspects of implementing this more powerful approach are quite tractable. The analysis itself is quite sophisticated, and makes use of powerful techniques for linear programming, but in the field the application of these techniques simply requires that the routing of any given container be permitted to depend on its readings up until that point. We believe that with a combination of centralized computers, wireless links to hand-held readers, and the backup information provided by bar codes attached to the containers, this routing and management can be easily handled with current logistic programming capabilities. We believe that implementation of this strategy, which makes much better use of the information available from sensors than do present schemes, will play a significant role in increasing the security of ports and of commerce.

# References

[1] Z. Chair and P.K. Varshney. Optimal data fusion in multiple sensor detection systems. *IEEE Trans. Aerospace Electron. Sys.*, **22** (1986), pp. 98–101.

[2] M. Cherikh and P.B. Kantor. Counterexamples in distributed detection. *IEEE Transactions on Information Theory* **38**(1) (1992), pp. 162-164.

[3] V. Chvátal, *Linear Programming*. Freeman, 1983.

[4] P.D. Stroud and K.J. Saeger, Enumeration of increasing Boolean expressions and alternative digraph implementations for diagnostic applications. Proceedings of Computer, Communication and Control Technologies (volume IV, H. Chu, J. Ferrer, T. Nguyen and Y. Yu, eds.) International Institute of Informatics and Systematics, Orlando, FL, 2003, pp. 328-333.

[5] P.C. Gilmore and R.E. Gomory, A Linear Programming Approach to the Cutting Stock Problem, Part I. *Operations Research* **9** (1961), pp. 849–859.

[6] P.C. Gilmore and R.E. Gomory, A Linear Programming Approach to the Cutting Stock Problem, Part II. *Operations Research* **11** (1963), pp. 863–888.

[7] R. Greiner, R. Hayward and M. Molloy, Optimal Depth-First Strategies for And-Or Trees. Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence (AAAI 2002), pp. 725-730.

[8] L.G. Khachiyan, A polynomial algorithm in linear programming, (in Russian). *Doklady Akedamii Nauk SSSR*, **244** (1979), pp. 1093–1096.

[9] H. Kushner and A. Pacut. A Simulation Study of Decentralized Detection Problem, *IEEE Trans. on Automatic Control*, vol. **AC-27**, No. 5, pp. 1116-1119, 1982.

[10] P.K. Varshney, Multisensor data fusion. *Electronics & Communication Engineering Journal*, **9** (1997), pp. 245-253.

# Appendix

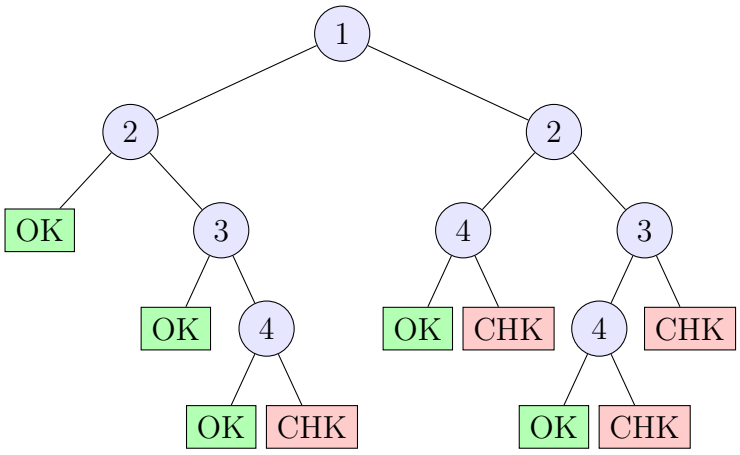


Figure 15: A decision tree representing second best pure strategy for four sensors with one threshold each.

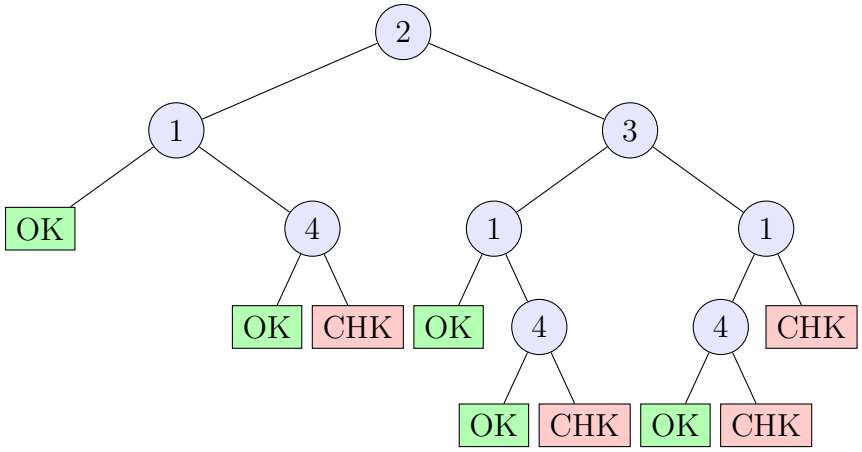


Figure 16: A decision tree representing third best pure strategy for four sensors with one threshold each.

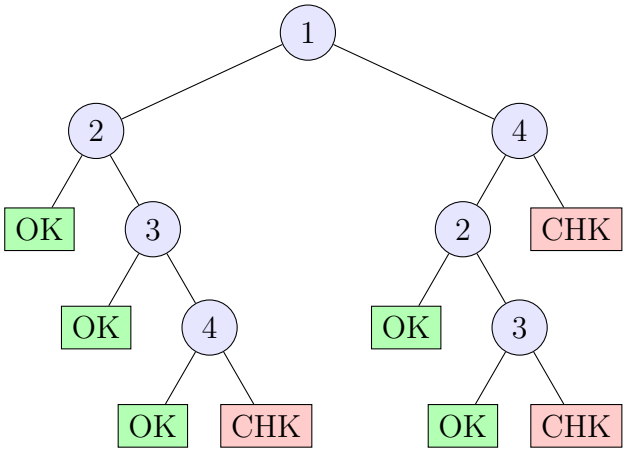


Figure 17: A decision tree representing forth best pure strategy for four sensors with one threshold each.

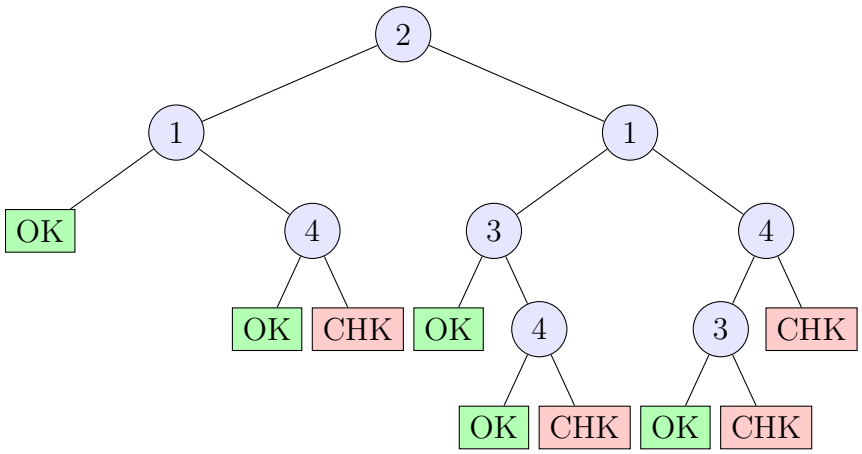


Figure 18: A decision tree representing fifth best pure strategy for four sensors with one threshold each.

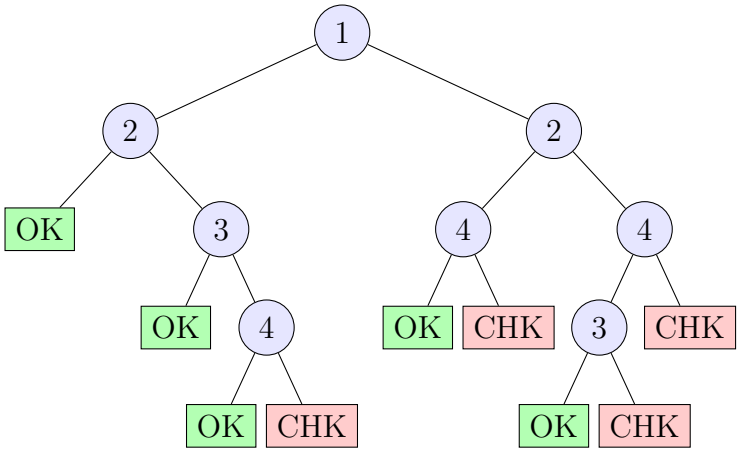


Figure 19: A decision tree representing sixth best pure strategy for four sensors with one threshold each.

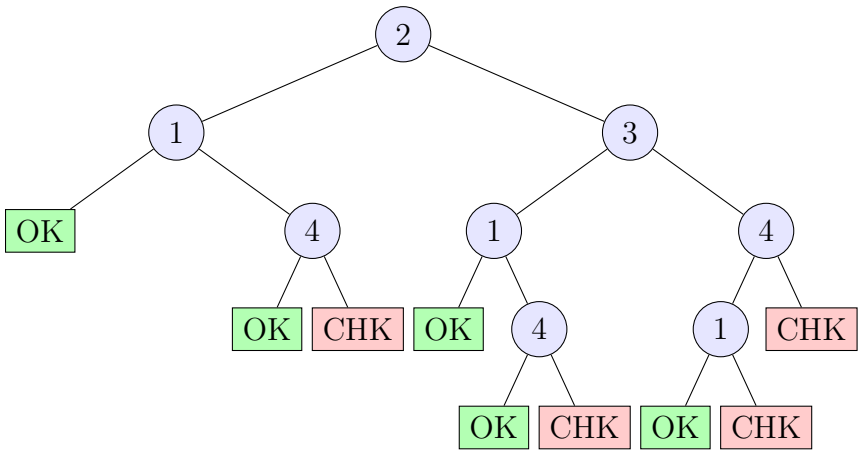


Figure 20: A decision tree representing seventh best pure strategy for four sensors with one threshold each.