

A dynamic programming algorithm for enumeration of efficient container inspection strategies

Noam Goldberg, Jonathan Word, Endre Boros, Paul Kantor

DyDAn - Center for Dynamic Data Analysis, Rutgers University, Piscataway, New Jersey

Introduction

In the container inspection problem we seek to maximize detection of a rare and hazardous cargo subject to a budget constraint. The best detection rate would be obtained by inspecting each and every container, but it is not feasible to do so. In order to satisfy a realistic budget constraint we can inspect only those containers flagged as suspicious in some sequence of tests (e.g. inspection of the manifest or sensor readings). The testing strategies are realized by a decision tree whose leaf nodes represent decisions to either completely examine, or to ‘release’ the container. We describe optimal testing strategies to be mixtures of decision trees as in (Boros et al. 2006).

There are no perfect testing strategies

A testing strategy is a mixture of decision trees whose nodes are tests (e.g. sensors) and the leaves correspond to decisions to ‘release’ or ‘inspect’.

- Nearly all testing strategies have false alarms:
- We denote by (f, d) the conditional probabilities of false alarm, for harmless items and of detection, for dangerous items, respectively.
- Testing strategies are costly:
- Assume the unpacking cost is 1 and the operating cost C_s is in [0,1].
- The cost of the interruption caused by stopping an item is K.
- Then the total cost per unit of examining an item using strategy s :

$$C_s(f) = C_s + \pi d + (1-\pi)f(1+K).$$

Note: Maximizing detection rate lets **decision makers** assess the tradeoff between inspection costs, and the risk of missing hazardous cargo.

The computation, however, assumes that π is negligible in comparison to 1.

The score given by each test is an indication of the probability that container is “dangerous”. Further testing, e.g. using additional sensors, may provide additional information until it is worthwhile to fully inspect an item.

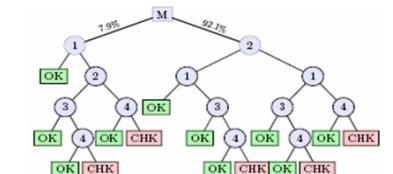


Figure 1. Mixtures of decision trees as an inspection strategy

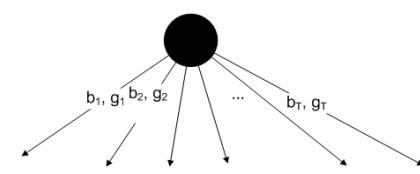


Figure 2. Zooming in on a node in the tree with an initial set of bins $\{(b_i, g_i)\}$

Assigning sensor readings to bins

When a sensor is discrete (e.g. document checks) the readings (or score) naturally define “bins”. When it gives continuous scores (as does a radiation detector) our approach is to discretize the score into a convenient number of bins. Each bin i will be characterized by the probabilities b_i, g_i that a dangerous (respectively, harmless) container falls into that bin.

Assigning bins in the space of scores

- a) break the range of possible scores into regions (“grid”)
- b) select one or more thresholds (at the grid points)
- c) Scores between adjacent thresholds are said to be “in the same bin”

Moving to ROC space

The score distributions define “odds ratio” for each region. The best detection for a given rate of false alarms is found by selecting regions, in the space of scores, in decreasing order of the odds ratio (by Neyman-Pearson Lemma). The ROC curve plots the resulting $d(f)$ - the detection rate as function of false alarm rate. We anticipate that fewer bins are needed, in ROC space, to obtain the same detection rate at a given false alarm rate.

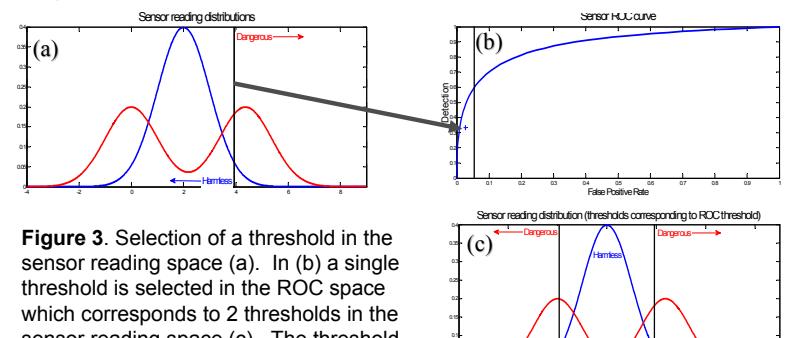


Figure 3. Selection of a threshold in the sensor reading space (a). (b) a single threshold is selected in the ROC space which corresponds to 2 thresholds in the sensor reading space (c). The threshold selected in (a) corresponds to a dominated point shown in (b).

From ROC to cost-detection curves

Cost detection curves show the detection rate as a function of total cost of a testing strategy. When comparing available testing strategies we may disregard testing strategies that are dominated (Fig. 4) in the cost-detection space. For example, if we are only allowed to use single sensor testing strategies, we do not need to consider sensors that have a lower detection rate for each budget values. Because strategies can be linearly mixed, cost-detection curves must be concave (as are ROC curves).

However, when considering the fusion of a sensor with a set of testing strategies using other sensors, we may need to use dominated sensors in the tree. For example consider the sensors s_1 and s_2 (Fig. 4). Here s_1 clearly dominates s_2 . In fact, s_1 not only dominates s_2 in the cost-detection space, but also has a lower operating cost per unit than s_2 .

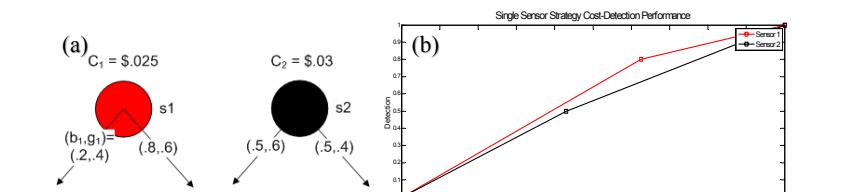


Figure 4. (a) Sensors s_1 and s_2 , with their associated costs and bins b_i, g_i . (b) The corresponding cost-detection curves of single sensor strategies are shown where it is evident that sensor s_1 dominates s_2 . The (efficient) strategies are constructed by attaching the strategies ‘release’ and ‘inspect’ to the left and right branches respectively.

When considering the testing strategies built from the set of two sensors then we may find that we may need to use s_2 . In fact we may need to use s_2 , the more expensive, dominated sensor as the root sensor for a wide range of budget values. (Fig. 3).

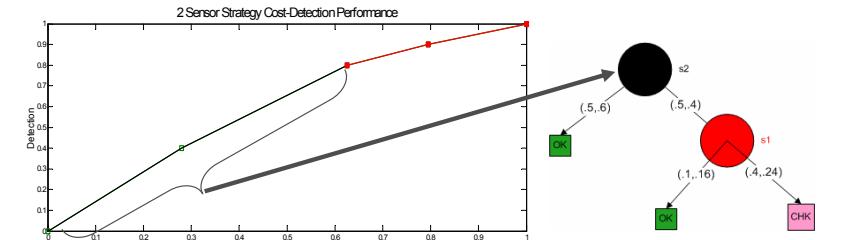


Figure 5. The cost-detection curve for two sensor free strategies using s_1 and s_2 . The vertex at cost of 0.28 corresponds to an optimal free with the dominated (and expensive!) s_2 as the root sensor. This example demonstrates that intuitive heuristics, such as using efficient inexpensive sensors, first do not necessarily work.

Dynamic Programming Overview

The problem is to find that strategy which gives the largest detection for rate, for any budget of interest. The problem is linear (see Boros et al.). We use DP to solve for the entire cost-detection curve at once.

Basic Facts:

- With a single (budget) constraint, every optimal mixed strategy is a mixture of at most 2 pure strategies.
- The curve of efficient strategies (efficient frontier) is a piecewise linear curve in cost-detection space consisting of the optimal strategies for each budget value.

Our Approach:

- Find the curve by enumerating its vertices – efficient *pure* strategies
- Make use of cost-detection dominance when possible
- The efficient frontiers using $k+1$ sensors are constructed from the set of frontiers using all subsets containing k sensors different from the one added.
- We will call the linking sub-problem of adding another sensor as a root above an existing set of (pure) strategies, the “sensor fusion problem”

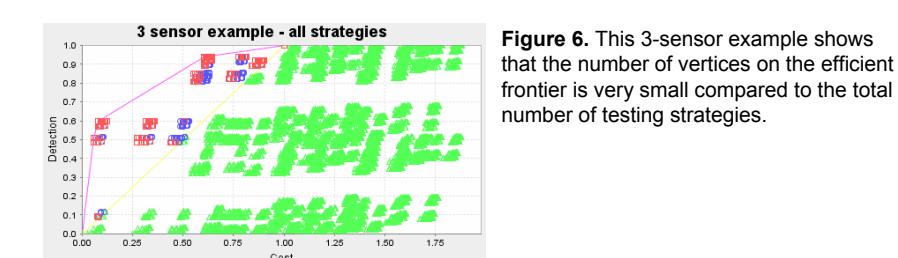


Figure 6. This 3-sensor example shows that the number of vertices on the efficient frontier is very small compared to the total number of testing strategies.

The sensor fusion problem

Given a set of available testing strategies $(C_1, D_1), \dots, (C_N, D_N)$ in cost-detection space, and a sensor with a set of bins characterized by $(b_1, g_1), \dots, (b_T, g_T)$:

- how to assign bins to the set of testing strategies in order to maximize detection?
- Each bin must be assigned to a testing strategy
- For a specific budget value the problem gives rise to a new special case of a Linear Multiple Choice Knapsack problem (Zemel 1980).
- Can be solved using a greedy algorithm in $O(NT \lg T + N \lg N)$ - for all values of budget M
- If the testing strategies are pre-sorted by cost, we can solve in $O(NT \lg T)$

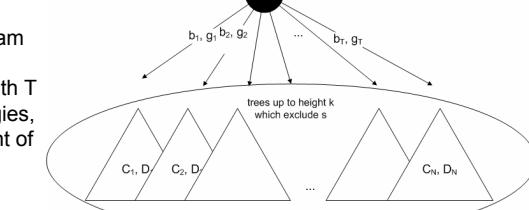


Figure 7. Schematic diagram showing the tree fusion problem: given a sensor with T bins, and a set of N strategies, what is the best assignment of strategies to bins.

$$\begin{aligned} D^{(\text{new})}(M) &= \max \sum_{i=1, j=1}^{i=T, j=N} b_j D_j^{(\text{lower})} x_{ij} \\ \text{s.t.} \\ \sum_{j} g_j C_j x_{ij} &\leq M \\ \sum_j x_{ij} &= 1 \quad \text{for each } i \\ 0 \leq x_{ij} &\leq 1 \quad \text{for each } i, j \end{aligned}$$

Dynamic Programming Formulation

When fusing sensor s as the root sensor, we consider only testing strategies that exclude s . Let $f^{(0)}(S)$ be the vertices of the efficient frontier, of trees up to height k , using sensors in S . The base case is simply the diagonal line connecting the ‘release all’ and ‘inspect all’ strategies. The DP iteration step is the following.

$$f^{(k+1)}(S) = \text{convhull}_{S \subseteq S} \left(f^{(k)}(S \setminus \{s\}) \cup \{(0,0), (1,1)\} \right)$$

In total, at most $2^{|S|}$ states and sensor-fusion problems need to be solved (Fig. 9). The *convhull* method has a linear runtime complexity if the sets are pre-sorted (as described for the *sensorfusion* method in the above).

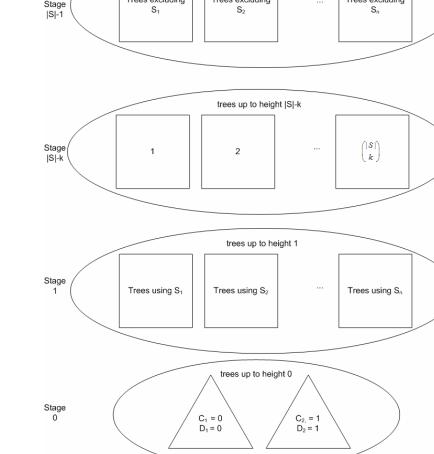


Figure 9. A schematic diagram showing the DP stages from bottom to top (shown as ellipses), and the states that correspond to each stage (squares).

| # of Bins | # of Sensors | Frontier Vertices | CCU Search | | |
|-----------|--------------|-------------------|------------|--------|--------|
| Min | Median | Max | Min | Median | Max |
| 2 | 2 | 3 | 0.00 | 0.00 | 0.43 |
| 3 | 3 | 5.5 | 9 | 0.03 | 0.63 |
| 4 | 4 | 8.5 | 17 | 0.08 | 0.13 |
| 5 | 5 | 11 | 33 | 0.23 | 0.29 |
| 6 | 6 | 24 | 44 | 0.61 | 1.34 |
| 7 | 14 | 51.5 | 94 | 1.68 | 2.78 |
| 8 | 16 | 80 | 252 | 3.81 | 9.475 |
| 9 | 17 | 94 | 481 | 8.31 | 19.16 |
| 10 | 44 | 279.5 | 758 | 32.93 | 117.99 |
| 3 | 2 | 4 | 10 | 0.00 | 0.00 |
| 4 | 3 | 15 | 26 | 0.00 | 0.08 |
| 5 | 7 | 26.5 | 79 | 0.13 | 0.25 |
| 6 | 14 | 70 | 162 | 0.55 | 1.65 |
| 7 | 38 | 272 | 625 | 2.54 | 11.025 |
| 8 | 2 | 3 | 8 | 0.00 | 0.02 |
| 9 | 4 | 32 | 57 | 0.07 | 0.14 |
| 10 | 23 | 87.5 | 241 | 0.38 | 0.965 |
| 8 | 24 | 382.5 | 737 | 1.39 | 44.675 |
| 9 | 2 | 5 | 11.5 | 0.01 | 0.03 |
| 10 | 17 | 65 | 119 | 0.16 | 0.36 |
| 11 | 4 | 300.5 | 483 | 1.74 | 6.098 |

Figure 10. Table showing an empirical study of the number of vertices and computation time of the dynamic programming algorithm. Each setting of number of bins and number of sensors (i.e. each row) corresponds to 20 runs generating random sensors. The experiments were run on an i686 3.40 GHz (x 2) 1014MB Linux machine.

Conclusions

- Assuming that the container tests are stochastically independent, we are able to formulate a **dynamic programming algorithm** that enumerates all efficient testing strategies, that is, a set whose members and mixtures dominate all other strategies in the cost-detection space.
- We find that the set of efficient testing strategies is substantially smaller than the set of all possible testing strategies.
- The number of *all* possible detection trees is at least double-exponential in the number of tests, the dynamic programming algorithm’s worst case runtime complexity is bounded by $O(2^N \text{poly}(S, T))$ (*the number of efficient strategies*).
- Experiments confirm that, in fact, the algorithm takes at most a few minutes, on a 3.4GHz machine with 1024MB of RAM, for the number of sensors that are currently being considered.

Future plans

- We plan to extend this research with
- computational improvements:
 - More efficient implementations
 - Theoretical approximation of the efficient frontier by a smaller efficient frontier, bounded by a polynomial in the input size.
 - extension to detection of cargo which is not as rare (that is, which has a non-negligible prior probability)
 - extension to detection of multiple types of contraband cargo
 - extensions to stochastically dependent sensors and re-measurements

Literature cited

- Boros E., Fedzhort L., Kantor P., Saeger K., Stroud P. (2006). *Large scale LP model for finding optimal container inspection strategies*. RRR 26-2006. http://rutcor.rutgers.edu/pub/rrr/reports2006/26_2006.pdf. To appear in Naval Research Logistics Quarterly.
- Jacobson S. H., Kamani T., Kobza J. E. (2005). *Assessing the impact of deterrence on aviation checked baggage screening strategies*. International Journal of Risk Assessment and Management, Vol. 5, No. 1, pp. 1-15.
- Kantor P., Boros E. (2007). *Deceptive detection methods for optimal security with inadequate budgets: the Screening Power Index*. RRR 26-2007. http://rutcor.rutgers.edu/pub/rrr/reports2007/26_2007.pdf.
- Madigan D., Mittal S., Roberts F. (2007). *Sequential decisions-making algorithms for port-of-entry inspection: Overcoming computational challenges*. Proceedings IEEE Intelligence and Security Informatics, New Brunswick, NJ. ISI2007, pp. 1-7.
- Zemel E. (1980). *The Linear Multiple Choice Knapsack Problem*. Operations Research, Vol. 28, No. 6, pp. 1412-1423.

Acknowledgments

This project was funded through the DyDAn Center of Excellence by a grant supported in part from the Department of Homeland Security, Science and Technology Directorate, Office of University Programs and by the Domestic Nuclear Detection Office (DNDO), of the Department of Homeland Security, through NSF Grant #CBET-0735910.

For further information

Please contact:
 Prof. Endre Boros, boros@rutcor.rutgers.edu
 Prof. Paul Kantor, kantor@scis.rutgers.edu
 Noam Goldberg, ngoldberg@rutcor.rutgers.edu
 Jonathan Word, jword@rutcor.rutgers.edu

This presentation can be found online at:
<http://rutcor.rutgers.edu/~ngoldberg/efficientstrategiesdhs08.pdf>

