AIIM

Association for Information and Image Management

1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

301/587-8202

Centimeter

Inches

MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.

1 of 1

Conf-941010--5

LA-UR- 94-1936

Title: RAOPS: Resource Allocation Optimization Program for Safeguards

Author(s): Andrew Zardecki and Jack T. Markin

Submitted to: IEEE International Carnahan Conference on Security Technology, Albuquerque, New Mexico, October 12-14, 1994

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

Los Alamos
NATIONAL LABORATORY

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

Form No. 836 R5
ST 2629 10/91

NIS-7/94-490
(March 1994)

Multiobjective Resource Allocation Problem for Safeguards

Andrew Zardecki

Safeguards Systems Group
Group NIS-7, MS E541
Los Alamos National Laboratory
Los Alamos, NM 87545

RAOPS—Resource Allocation Optimization Program for Safeguards—is extended to a mulitiobjective return function having the detection probability and expected detection time as criteria. The expected detection time is included as a constraint, based on the well-known Avenhaus model of the optimum number of inventory periods. Examples of computation are provided.

1. Introduction

Selecting the safeguards elements that constitute a system for protecting special nuclear materials (SNM) is a complex process involving choices about those technologies and procedures that are most effective in countering a range of threats. The computer program RAOPS (Resource Allocation Optimization Program for Safeguards), developed at Los Alamos by the Safeguards Systems Group,[1] is a tool for aiding an analyst in the design of a safeguards system for either new facilities or for upgrades of existing facilities. For a serial or divergent arrangement of activities, the program determines the configuration of safeguards options that maximizes the detection probability against a range of scenarios for theft or diversion of SNM under the constraint of fixed safeguards resources. Here, the term "activity" refers to an area or boundary in which one or more safeguards elements can be deployed.

RAOPS is an example of the resource allocation problem, in which the overall detection probability is the single objective function. Other examples of this type of single-criterion optimization problem include resource allocation problems with fuzzy detection probabilities[2] and the defense in depth problem, solved by stochastic optimization.[3] When we want to maximize the detection probability and minimize the detection time, we face the multiobjective optimization problem.[4]

In this report, we first summarize the essential features of the initial and final state problems in dynamic programing. The solutions to these two problems are the backward and forward recursion, respectively.[5] To minimize the expected detection time, while maximizing the detection probability, we incorporate the expected time as a constraint on paths in the state space.[6] Finally, we comment on a possible extension of the formalism to a nonserial configuration of activities.

2. *Initial and final state problem in dynamic programming*

The serial configuration corresponds to the situation in which the adversary sequentially encounters each safeguards activity. This has been studied earlier by Markin, et al.,[7] and by Fishbone.[8] Figure 1 shows a typical arrangement, with the corresponding tree structure depicted in Fig. 2.

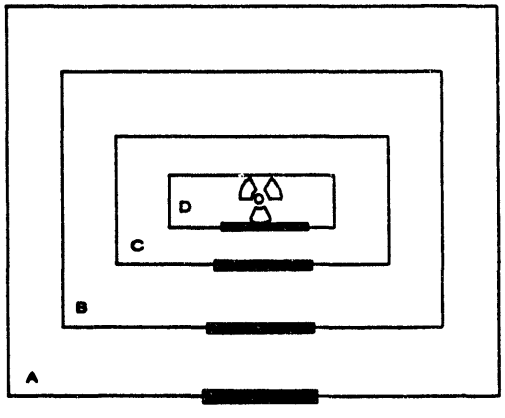

Fig. 1. Serial configuration of four activities.

Fig. 2. The tree (list) structure of activities corresponding to Fig. 1.

In practice, though, a realistic arrangement will involve multiple choices; for example, after penetrating the main facility gate A, followed by the inner fence B, the adversary has to select one of the two different storage areas. Figures 3 and 4 may be contrasted with Figs. 1 and 2, respectively. Stated differently, a nonserial configuration implies a structure in which there are branch points in the adversary's path to a goal.

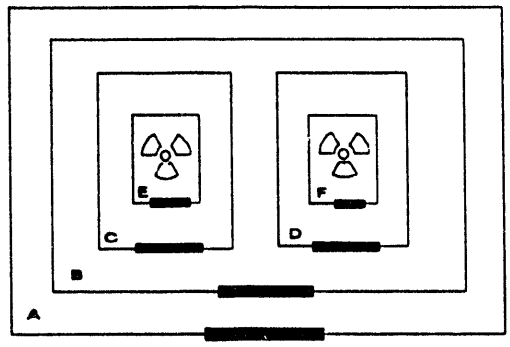

Fig. 3. Divergent configuration of activities.

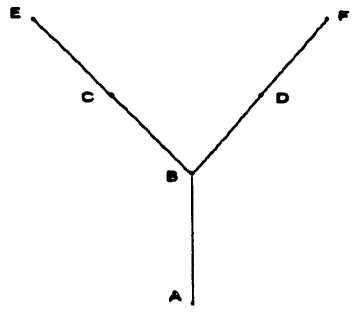

Fig. 4. The tree structure of activities correspond- to the configuration of Fig. 3.

A word about the terminology used here is in order. In the context of safeguards, when maximizing the detection probability, we speak of the return function; on the other hand, when we minimizing the nondetection probability, we minimize the cost function. In either case, we optimize the objective function.

2.1 Discrete Dynamic Programming Algorithm

We start by listing some basic results and by explaining our notation. For a serial system, the multiplicative return (objective) function is

$$R = \prod_{k=1}^{N} r[x(k), u(k), k] \ . \tag{1}$$

Here $x(k)$ and $u(k)$ refer to state variables and decision variables at stage k. See Fig. 5. The system equations describe how the state variables at stage $k+1$ are related to the state variables at stage k. These equations are written as

$$x(k+1) = g[x(k), u(k), k] \ , \tag{2}$$

where g is a known function.

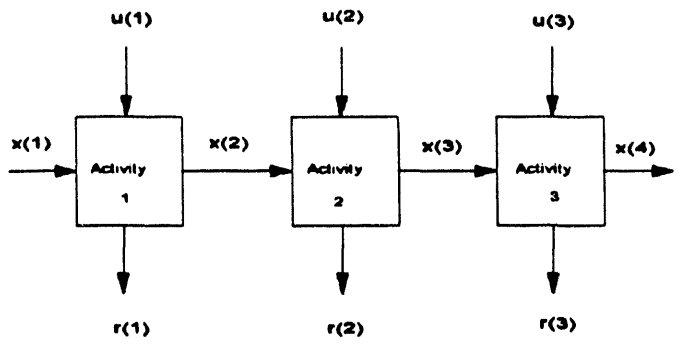

Fig. 5. Serial multistage decision system. Shown are state variable x, decision variable u, and the return (objective) function r at each stage.

For the resource allocation problem, g is simply a difference of x and u:

$$x(k+1) = x(k) - u(k) , \tag{3}$$

and the function $r[x(k),u(k),k]$ specifies the nondetection probability at stage k.

The dynamic programming optimization solves the following iterative functional equation for the optimum return $I(x,k)$

$$I(x,k) = \min_{u \in U} \{ r(x,u,k) \cdot I[g(x,u,k), k+1] \} \tag{4}$$

for $k = 1, ..., N-1$, by minimizing the expression in the curly brackets over the set U of decisions. The initial condition is

$$I(x,k) = \min_{u \in U} \{ r(x,u,N) \} . \tag{5}$$

This recursion procedure, called backward recursion, solves the initial state problem, in which the optimal N-stage return becomes a function of the input to stage one. When state inversion is possible, as in the resource allocation problem, one can also use forward recursion to solve a final state problem. In the final state problem, the optimal return is found as a function of the stage output. Finally, the initial-final state optimization consists in finding the optimal return as a function of the input to stage one and the output from the last stage.

2.2 *Initial State Problem*

A serial multistage system consists of a set of stages joined together in series so that the output of one stage becomes the input to the next (see Fig. 5). As indicated in Eq. (1), the return function R is a product of the return functions referring to individual stages.

In the initial state problem, we determine the optimum value of R as a function of the state variables $x(1)$ at the first stage. The initial state problem is solved by the backward recursion according to the scheme indicated in Fig. 6. Here the discretized state variable x is plotted versus the stage index k. The decision variable u is also assumed to be discretized.

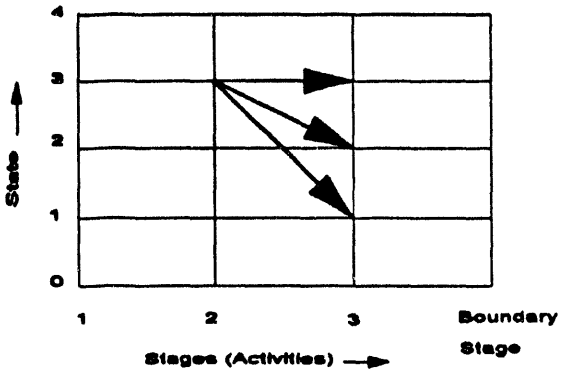

Fig. 6. The backward recursion scheme.

In the backward recursion, we decide which state to move "to" in stage k, given a state at stage $k+1$. For n stages, introducing the boundary conditions at the fictitious stage $n+1$, we first decide at stage n to which state we move to attain the optimal objective. We then proceed backwards, arriving eventually at stage one.

2.2.1 Final State Problem

In the final state problem, we determine the optimal solution as a function of boundary conditions at the nth stage. Using forward recursion,[9] we decide which state to move "from" in stage k, given a state in stage $k+1$.

For n stages, we arrive at the fictitious end stage $n+1$, at which point we have solved the final state problem; the optimal return is expressed as a function of the remaining resources at stage $n+1$.

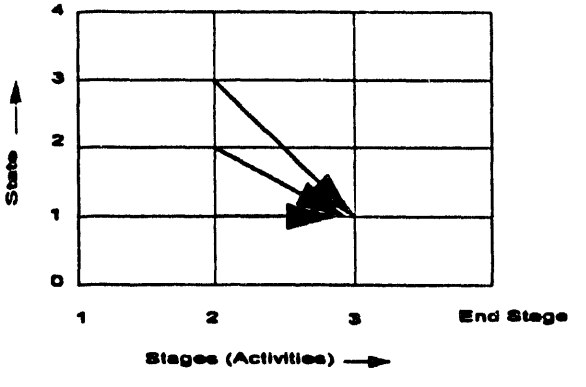

Fig. 7. The forward recursion scheme.

3. Detection Time as a Constraint

Given the fixed amount x_0 of the resources (budget) at stage one, the forward recursion defines $x_0 + 1$ optimal paths $\ell(i)$, $i = 0, ..., x_0$ in the state-stage plane. Each path starts at the point $(1, x_0)$ and terminates at the point $(n + 1, i)$, determined by the remaining budget i. Figure 8 illustrates the situation for the initial budget $x_0 = 4$.

We introduce the expected detection time by following the procedure described by Avenhaus[10] in the context of the optimum number of inventory periods. If the time needed to reach the kth activity is T_k, then the expected detection time T is given as

$$T = \frac{\prod_{k=1}^{n} T_k P_k \prod_{i=1}^{k-1} (1 - P_i)}{\prod_{k=1}^{n} (1 - P_i)} \tag{6}$$

Here, P_i is the detection probability, regarded as the optimal detection probability determined through the final-state optimization.

A supplementary benefit of the forward dynamic programming solution is that it spawns trajectories reaching different terminal states. To each trajectory, defining the total return with a given amount of remaining resources, a terminal constraint can be applied. If other criteria—such as the expected detection time—are considered, the trajectory satisfying these supplementary criteria can be selected.

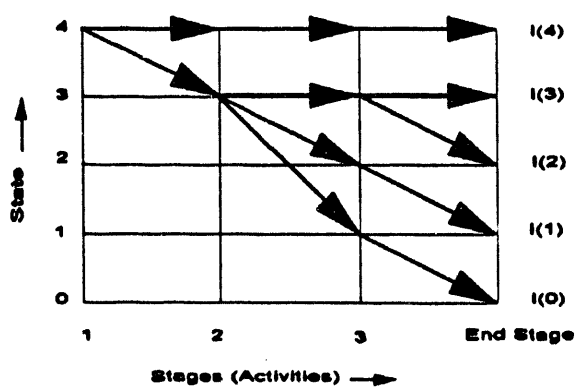

Fig. 8. Optimal paths.

When dealing with a divergent configuration of activities, viewed as a tree data structure, we reduce the problem to a serial configuration, according to the scheme shown in Fig. 9.

In the example of Fig. 9, backward dynamic programming is applied first to activities labeled 3 and 4 in the upper branch of the tree. This yields the optimal resource allocation for the upper branch. A similar procedure is used for activities 5 and 6 of the lower branch of the tree. Finally, the minimax (or equivalent) procedure combines the two branches into a node named B.

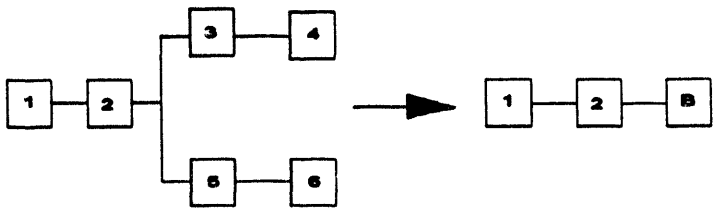

Fig. 9. Transformation of a serial configuration of activities.

The simplest way to include the expected detection time into the optimization process is to consider, for example, the largest time one needs to reach any of the nodes in the tree branches. With this time assigned to node B, the problem is reduced, then, to the problem of minimizing the expected detection time. A more sophisticated procedure would weight various times required to reach the nodes in the upper and lower branches.

When using the forward dynamic programming algorithm with a fixed initial amount of resources, there is a combinatorial explosion of the paths that can be drawn in the case of a complicated tree data structure. The crucial point is to consider only selected paths. We achieve this objective by applying backward dynamic programming to the activities belonging to the branches of the activities tree.

4. *Example of Computation*

As an illustration of our considerations, we use a simple arrangement of activities shown in Fig. 1. Table 1 lists the safeguards options, their associated costs, and times needed to reach each activity. A total budget of 10 units is assumed.

TABLE 1. Sample data for a serial configuration of activities shown in Fig. 1.

Activity	Option's Cost	Option's Detection Probability	Time to reach activity
A	1	0.2	1
	2	0.4	
B	1	0.4	2
	2	0.6	
C	1	0.3	3
	3	0.6	
D	3	0.7	4
	4	0.9	

The solution to the backward recursion scheme is displayed in Fig. 10, in which the detection probability is shown as a function of the available budget. It is evident that, beyond a certain budget, the detection probability does not increase fast enough to justify an additional investment.

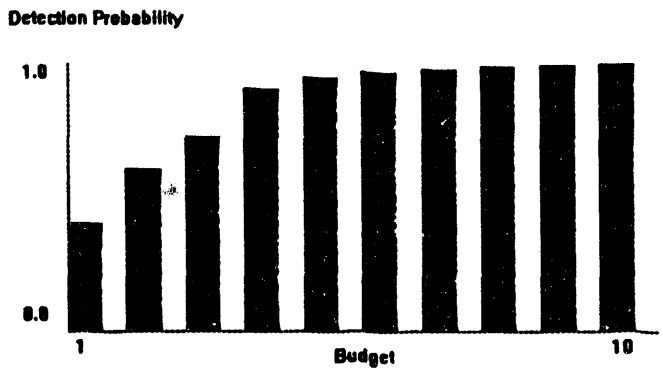

Fig. 10. Detection probability as a function of available budget.

When the expected detection time is evaluated, the solution—obtained by the forward recursion method—is shown as a function of the remaining resources. This solution is depicted in Fig. 11. In Fig. 12, we show the expected detection time as a function of available budget.

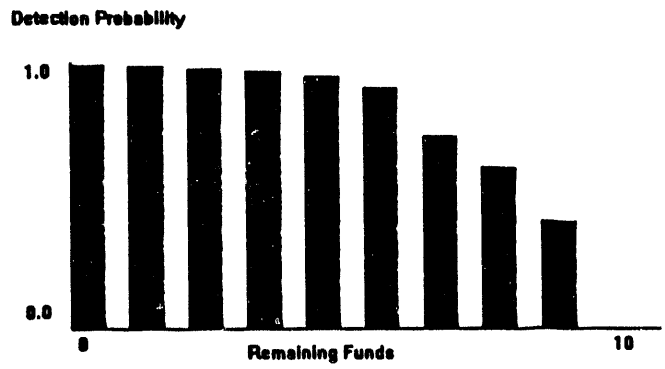

Fig. 11. Detection probability as a function of the remaining funds.

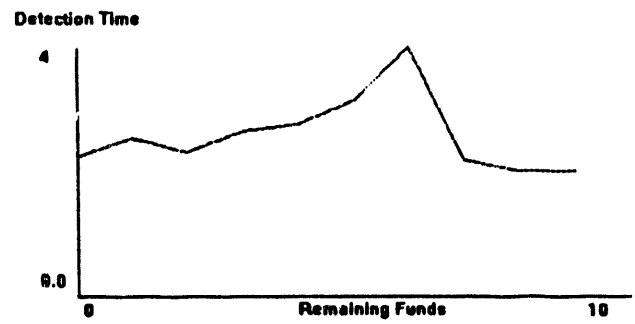

Fig. 12. Expected detection time as a function of the remaining funds.

As should be intuitively clear, the detection probability decreases as the remaining funds accumulate. The expected detection time displays a maximum of seven for the remaining funds; it is reasonably small when four units of the initial budget remain. This implies that, with the total budget of 10 units, the allocation of six units leads to the highest detection probability and a reasonable value of the expected detection time.

5. Conclusions

We have implemented the multicriteria resource allocation problem with the overall detection probability and expected detection time viewed as the joint return function. Whereas the backward recursion is the natural tool for solving the standard resource allocation problem, the forward recursion is better adapted to the multiobjective allocation.

The emphasis of this paper was placed on the expected time needed to reach the target; future research will focus on the time needed by an adversary to leave the facility after committing a diversion.

References

1. A. Zardecki, J. T. Markin, and P. Henriksen, "Resource-Allocation Optimization Program for Safeguards," Los Alamos National Laboratory report LA-12244-M (March 1992).

2. A. Zardecki and J. T. Markin, "Fuzzy Resource Optimization for Safeguards," in *Proceedings of the Fourth International Conference on Facility Operations-Safeguards Interface* (American Nuclear Society, Inc., La Grange Park, Illinois, 1991), pp. 443–447.

3. A. Zardecki and J. T. Markin, "Defense in Depth and Resource Allocation for Safeguards," *Nucl. Mater. Manage.* **XX**, 426–429 (1991).

4. C-L. Hwang and A. S. Masud, *Multiple Objective Decision Making—Methods and Applications* (Springer, Berlin, 1979).

5. G. L. Nemhauser, *Dynamic Programming* (Wiley, New York, 1966).

6. R. E. Larson and J. L. Casti, *Principles of Dynamic Programming* (Marcel Dekker, New York, 1978).

7. J. T. Markin, "Resource Allocation Optimization Program for Safeguards," Los Alamos National Laboratory, Safeguards Systems Group report N-4/89-422 (1989).

8. L. G. Fishbone, "Optimal Allocation of Safeguards Inspection Resources for a Nuclear Fuel Cycle," Brookhaven National Laboratory report BNL-34127, December 1973.

9. D. J. White, *Dynamic Programming* (Holden-Day, San Francisco, 1969).

10. R. Avenhaus, *Material Accountability: Theory, Verification, and Applications* (Wiley, New York, 1977).

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DATE
FILMED
10/25/94

END