FACULTY WORKING PAPER 92-0166

Optimal Sequential File Search

George E. Monahan
Department of Business Administration

Bureau of Economic and Business Research
College of Commerce and Business Administration
University of Illinois at Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 92-0166

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

September 1992

Optimal Sequential File Search

George E. Monahan

Department of Business Administration

Optimal Sequential File Search[1]

GEORGE E. MONAHAN

Department of Business Administration
University of Illinois at Urbana-Champaign
Champaign, IL 61820, USA

Final Revision: September, 1992

Abstract: This paper analyzes a dynamic problem in decision-making under uncertainty in which the decision-maker must determine whether or not to purchase costly information. The objective is to see whether or not a record is in a sequential computer file whose contents are initially not known. It is possible to gather information about the contents of the file by examining one or more of its positions. Obtaining this information is costly, however. The problem of determining an optimal search and disposition strategy is formulated as a Markov decision process. A finite-step algorithm for computing an optimal policy is presented. Finally, qualitative properties of the optimal search procedure are discussed. One particularly interesting result establishes conditions under which a "search-while-uncertainty- persists" strategy is optimal.

Keywords: Search, decision, Markov decision processes, optimization

[1]Forthcoming, *European Journal of Operational Research*

Optimal Sequential File Search

GEORGE E. MONAHAN

Department of Business Administration
University of Illinois at Urbana–Champaign

1. Introduction

This paper analyzes a dynamic problem in decision-making under uncertainty in which the decision-maker must determine whether or not to purchase costly information. A sequential computer file consists of records that have been randomly selected from a known, finite population of records. The selected records are stored in sequential order on the basis of the value of some key that is contained in each record. A request is made regarding the status of a particular record in the population. The *sequential file search* problem is to determine the status of the requested record. Is the record in the file? If it is not, where should it be placed so that the sequential order is preserved? It is possible to gather information about the contents of the file by examining one or more of its positions. Obtaining this information is costly, however. Based upon all of the information that is available, a search strategy specifies the positions of the file to examine and a disposition strategy specifies whether or not the requested record is in the file and if it is not where it would placed if it were added to the file. A reward is earned only if the status of the record is correctly determined. An optimal search and disposition strategy maximizes the expected value of this terminal reward less the total cost of acquiring information.

If, for example, the cost of search relates to the time required to determine the status of a record in the population, the optimal search and disposition strategy then specifies how the file should be searched so as to minimize the expected length of time required to determine whether or not the requested record is in the file. Wiederhold (1977) describes several techniques for searching a sequential file but makes no assumptions regarding the likelihood of certain records being in the file. The model developed and analyzed here can be viewed as a Bayesian version of the binary search and probing schemes he discusses. The relationship between the problem studied here and other search problems in the computer science literature is discussed in the next section.

A more specific statement of the sequential file search problem is now given. There are n records in the population, labelled by "keys" that we assume are real numbers $r_1, \ldots, r_n$.

For convenience, positions within the file are called "boxes" and are labelled Box 1, ..., Box n. Nature determines the contents of the file: record r_i is in one of the n boxes with probability p_i and its inclusion is independent of the inclusion of other records in the population. Let N be the random variable that denotes the total number of records stored in the n boxes. These N records that actually constitute the file are ordered according to their keys: if i_j denotes the subscript of the jth record in the file, $j = 1, \ldots, N$, then record r_{i_j} is in Box j and $r_{i_1} \leq \ldots \leq r_{i_N}$. We wish to determine if some record, say r_k for some $1 \leq k \leq n$, is in the file. Initially, all that we know about the file and its contents are the probabilities p_i, $i = 1, \ldots, n$ that the various records are in the file. To assist us in the determination of the status of r_k, we can gather additional information regarding the contents of the file by examining the contents of any or all of the boxes. The cost to examine one box and determine its contents with certainty is \$$c$. Without loss of generality, we assume $0 < c < 1$.

After gathering information, we must declare that either r_k is in the file or it is not. If we declare that record r_k is not in the file, we must also indicate where in the file it would be added. In other words, we must indicate what "gap" the record is in. We say record r_k is in gap j if $i_j < k < i_{j+1}$, for some $j = 0, \ldots, n-1$. Without loss of generality, we receive a reward of \$1 if we are correct in our assessment regarding the status of r_k. If we are incorrect, we receive nothing. (Any values could be chosen for the rewards and/or costs associated with making both correct and incorrect decisions. The values used here are particularly convenient.) The objective is to determine both a search strategy that tells us which boxes to examine, as well as a disposition strategy that specifies either that the record is in the file or the gap in which it falls. An optimal search and disposition strategy is one that, given the initial probabilities regarding the contents of the file, maximizes the expected terminal payoff less total search cost.

As in most problems concerning sequential decision-making under uncertainty, the optimal search and disposition strategy seeks to achieve an economic balance between short and long-term rewards. The acquisition of information is costly, but better information improves the likelihood that the disposition decision results in higher expected payoffs. In general, these strategies are complex functions of all the information that is available at a given stage of the decision process. We model the search problem as a Markov decision process (MDP) and show how to compute optimal expected payoffs and an optimal

search and disposition strategy. The model formulation allows us to do a fairly extensive analysis of how these payoffs and strategies depend upon parameters of the problem. One particularly interesting result we establish is that under certain conditions, if it is initially optimal to examine one box, then it is optimal to continue the examination of boxes until all uncertainty has been resolved.

The paper is organized as follows. Section 2 relates the problem posed here with other forms of search problems in the economics and operations research literature and with other file search problems in the computer science literature. Section 3 presents the MDP model of the sequential file search problem and introduces a small numerical example that is used throughout the paper for illustration purposes. A solution procedure for solving the MDP is given in Section 4. Section 5 analytically derives several results that show how certain parameters of the problem influence both the optimal search and disposition strategy, as well as the optimal payoff function. Concluding remarks are in Section 6.

2. Related Literature

There is an extensive literature in both economics and operations research dealing with the search for hidden objects under a variety of informational assumptions. See, e.g., Stone (1975), for references to a large portion of this literature. In much of this literature, the objective is to examine boxes when the contents are unknown. The primary distinction of the problem analyzed in this paper is the inter-dependence resulting from the assumption that the contents of the boxes are completely ordered. There are both positive and negative ramifications associated with this inter-dependence. It is possible, for example, to obtain information concerning the contents of Box 1 by examining Box 2, a characteristic that can be beneficially exploited. On the other hand, the dependence rules out the optimality of "reservation price" strategies, which are relatively simple rules for determining which boxes to examine and when to terminate search. See Weitzman (1979) for an interesting discussion of the optimality of such rules in a particular class of search problems.

The sequential file search interpretation given in the Introduction can be viewed as the converse of the "typical" file search problem widely discussed in the computer science literature. In the classical problem (see, e.g., Knuth (1973)), a finite universe of possible records are linearly ordered with respect to a specified key. A sequential file contains a

known subset of records from that universe. These records are stored on the basis of their key values. The objective is to determine whether an object drawn from the universe is contained in the file. This determination can only be done by comparing the selected record to the known records already in the file. In the probabilistic version of the problem, the probability that an object that is randomly drawn from the population corresponds to a particular element of the file is known for all records in the file. The probability that the drawn object lies between any two records in the file is also specified. Knuth (1971) gives a dynamic program that specifies a strategy for minimizing the number of comparisons needed to determine the status of the record.

A decision-theoretic version of the classical file search problem is given as an example in Whinston and Moore (1986, 1987). Moore et al. (1988) analyze this problem in more detail and present a solution technique that is based upon the analysis of a decision tree. The problem studied in this paper is somewhat more complex than the classical search problem. For example, binary search schemes are common techniques for searching a file when the contents are known: starting at the middle of the file, half of the file is eliminated from consideration on the basis of the comparison of key values. Such a scheme cannot typically be employed for the file search problem studied here, however. Since we do not know how many records are in the file, we don't even know where the middle is! Beginning a search at the $[n/2]$th position in the file is certainly not uniformly optimal since it ignores the probabilities of records being the file. In Section 5, we discuss the optimality of such a strategy for a special case, however.

In the problem studied in Moore et al. (1988), there is uncertainty related to a single item, the object that is drawn from the universe. In the problem considered here, there is uncertainty regarding the entire contents of the file. While both problems entail the sequential acquisition of information, the methodologies used to determine optimal behavior differ. The state space in Moore et al. are subsets of key values that have not yet been excluded as a result of the search process. Here the state space is the set of probability distributions over the set of keys of the file.

The model developed here uses the fact that the contents of the file are stored in linear order but, for expositional purposes, does not fully exploit this feature of the problem. Blair and Monahan (1990) show how the state space of the model developed here can be greatly reduced, thus diminishing the computational effort required to generate an optimal

strategy. The reduction is somewhat cumbersome, however, and makes it considerably more difficult to establish qualitative properties of the optimal search and disposition strategy. In the next section, the file search problem is formulated as a Markov decision process.

3. A Markov Decision Process Model

Let $a_1, \ldots, a_n$ denote the contents of Boxes 1 through n, respectively. If Box j is empty, we say $a_j = \emptyset$. We define the "null record" as $r_{n+1} = \emptyset$, so that by convention, $\emptyset \geq r_j$ for all j. If $a_j = \emptyset$, then $a_i = \emptyset$, for $i = j+1, \ldots, n$. If there are m records in the file, then Boxes $1, \ldots, m$ each contain a record and Boxes $m+1, \ldots, n$ are empty. The contents of the file is described by exactly one of the elements of $\mathcal{B}$, where

$$\mathcal{B} = \{\mathbf{a} = (a_1, \ldots, a_n) \,|\, a_j \in \{r_1, \ldots, r_n, \emptyset\} \text{ for all } j \text{ and } a_i \leq a_{i+1} \text{ for } i = 1, \ldots, n-1\}.$$

Given there are n distinct potential records and multiple copies of a record are not permitted in the file (except, of course, for r_{n+1}), there exactly 2^n elements in $\mathcal{B}$. For convenience, let $\mathcal{C} = \{1, \ldots, 2^n\}$ index the 2^n vectors in $\mathcal{B}$. (Just how the elements in $\mathcal{B}$ are indexed is not important.)

We know that the contents of the file is specified by one of the 2^n elements in $\mathcal{C}$ but we don't know which one. For this reason, we refer to the states in $\mathcal{C}$ as (unobservable) *core* states. Since we know the probability that a particular record is in the file, we can compute the probability that each core state prevails. As we gather information regarding the contents of some of the boxes, we update the distribution over the core states via Bayes' rule. Let $\mathcal{S} = \{\Pi = (\pi_1, \ldots, \pi_{2^n}) \,|\, 0 \leq \pi_i \leq 1, \text{for } i = 1, \ldots, 2^n \text{ and } \sum_i \pi_i = 1\}$ be the set of probability mass functions defined on $\mathcal{C}$. We denote the *state of the decision process* at stage t as $\Pi_t \in \mathcal{S}$; that is, Π_t summarizes all of the information that is relevant for making decisions at stage t. (The MDP being formulated here is actually called a *partially observable Markov decision process* (POMDP), reflecting the fact that the core states are not directly observable; for a discussion regarding the sufficiency of Π_t as a state descriptor in POMDP's, see Monahan (1982).) At the beginning of stage t, we know the value of Π_t and can take actions that fall into two categories. First, we can terminate the search process and choose one of several stop actions: we can either declare that the record for

which we are searching is in the file or we can declare that the record is not in the file and falls in gap l, $l = 0, \ldots, n-1$.

If we decide not to stop the search process, we must choose which of the n boxes should be examined next at a cost of $\$c$. After examining the box, the probability distribution Π_t is updated via Bayes' rule to Π' (say) to reflect the information obtained. The decision process then moves to then next stage with the state specified by Π' and proceeds as it did at stage t.

3.1 Problem k

"Problem k" is to determine whether or not r_k is in the file for some $k = 1, \ldots, n$. We know that r_k is at most the kth "largest" record in the file and hence can never be Box $k+1, \ldots,$ Box n. Furthermore, an examination of the boxes whose numbers exceed k will not provide any useful information regarding the status of r_k. Therefore, in Problem k, we restrict our attention to the first k boxes.

Let B_{ij} denote the contents of Box j when the core state is i, for $j = 1, \ldots, k$ and $i \in \mathcal{C}$. Let $B_{i0} \equiv \emptyset$ for all i. The following sets are useful in the specification of the optimization problem. For notational convenience, we suppress the dependence of these sets on k:

$$S = \{i \in \mathcal{C} \mid r_k = B_{ij}, \text{ for some } j = 1, \ldots, k\}$$

$$G_l = \{i \in \mathcal{C} \mid r_k > B_{il} \text{ and } r_k < B_{i,l+1}\} \quad \text{for } l = 0, \ldots, k-1$$

$$H_{jq} = \{i \in \mathcal{C} \mid r_q = B_{ij}\} \quad \text{for } n+1 \geq q \geq j \text{ and } j = 1, \ldots, k.$$

S is the set of values that index core states that have r_k as one of their n elements; i.e., if $i \in S$, then r_k is in the file. For some index value l, $l = 0, \ldots, k-1$, G_l is the set of core states for which r_k is in gap l. Finally, H_{jq} is the set of values that index core states such that r_q is in Box j.

Notice that the sets S, G_l, and H_{jq} are determined by the data of the problem and are independent of the decision process; i.e., these sets are not influenced either by the state or the action taken in any state of the decision process.

In the next subsection we develop the functional equation of a dynamic program that specifies the optimal expected reward as a function of the state of the decision process. We then give a small numerical example that illustrates the various elements of the model.

3.2 Updating the State Vector via Bayes' Rule

Suppose we examine Box j and observe r_q, for $j = 1, \ldots, k$ and $n+1 \geq q \geq j$. With the definition of H_{jq}, we define the probability that r_q is in Box j as

$$\sigma_{jq}(\Pi) \equiv P\{r_q \in \text{ Box } j\} = \sum_{i \in H_{jq}} \pi_i. \tag{1}$$

Let

$$\xi_i^{jq} = \begin{cases} 0 & \text{if } i \notin H_{jq} \\ \pi_i & \text{if } i \in H_{jq} \end{cases}.$$

Then, by Bayes' rule, the posterior distribution incorporating the information that r_q is in Box j, given that Π is the prior distribution, is

$$\Xi_{jq}(\Pi) = \frac{1}{\sigma_{jq}(\Pi)}(\xi_1^{jq}, \ldots, \xi_{2^n}^{jq}). \tag{2}$$

Therefore, if Π is the state of the decision process and r_q is observed in Box j, the decision process moves to the state $\Xi_{jq}(\Pi)$.

3.3 The Optimal Value Function

We now define an infinite-horizon stochastic, dynamic program that specifies the optimal expected rewards that can be generated for any distribution over $\mathcal{C}$. An infinite planning horizon is used since there is no exogenous requirement that the search process stop. Since search costs are positive (and there are only a finite number of boxes to search), an optimal policy will not specify that searching be done indefinitely.

Let $V(\Pi)$ denote the optimal expected reward that can be earned over an infinite planning horizon when the state of the process is $\Pi \in \mathcal{S}$. This function, called the *optimal value function*, satisfies the following dynamic programming recursion:

$$V(\Pi) = \max \begin{cases} \sum_{i \in S} \pi_i & \text{expected reward if we stop and declare } r_k \text{ is in the file} \\ \sum_{i \in G_l} \pi_i & \text{expected reward if we stop and declare } r_k \text{ is in Gap } l,\ l = 0, \ldots, k-1 \\ J(\Pi, j),\ j = 1, \ldots, k & \text{expected reward if box } j \text{ is opened and the process proceeds optimally} \end{cases} \tag{3}$$

where

$$J(\Pi, j) = -c + \sum_{q=j}^{n+1} V\left(\Xi_{jq}(\Pi)\right) \cdot \sigma_{jq}(\Pi). \tag{4}$$

In (4), note that if we observe r_q in Box j (which happens with probability $\sigma_{jq}(\Pi)$), the new state is $\Xi_{jq}(\Pi)$) and the optimal expected payoff that can be generated over the remainder of the infinite planning horizon is $V(\Xi_{jq}(\Pi))$. The sum on the right-side of (4) is therefore the optimal expected payoff that can be obtained if one of the k boxes is opened, the contents observed, the state updated via Bayes' rule, and the decision process proceeds in an optimal manner.

3.3 A Numerical Example for $n = 3$

Suppose there are only three boxes. Let $r_1 = 1$, $r_2 = 3$, $r_3 = 9$, $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$, and $c = 0.1$. Then $\mathcal{B}$, the set of vectors denoting all of the $2^3 = 8$ possible values for the contents of the three boxes, is

$$\mathcal{B} = \{(\emptyset,\emptyset,\emptyset),(1,\emptyset,\emptyset),(3,\emptyset,\emptyset),(9,\emptyset,\emptyset),(1,3,\emptyset),(1,9,\emptyset),(3,9,\emptyset),(1,3,9)\}.$$

In this case, $\mathcal{S} = \{\Pi = (\pi_1,\ldots,\pi_8) : 0 \le \pi_i \le 1,\ i = 1,\ldots,8 \text{ and } \sum_{i=1}^{8} \pi_i = 1\}$.

The eight core states and initial probabilities that constitute Π_1, the initial state of the decision process, are numbered and listed in Table 1.

Core State	Contents of Box 1	2	3	Π_1
1	$\emptyset$	$\emptyset$	$\emptyset$	0.504
2	1	$\emptyset$	$\emptyset$	0.056
3	3	$\emptyset$	$\emptyset$	0.126
4	9	$\emptyset$	$\emptyset$	0.216
5	1	3	$\emptyset$	0.014
6	1	9	$\emptyset$	0.024
7	3	9	$\emptyset$	0.054
8	1	3	9	0.006

TABLE 1. Core States and Initial Probabilities

Suppose we wish to determine if "3" is in the file. Then $k = 2$ since $r_2 = 3$ in this example. The set of core state vectors that have "3" as one of their elements is $S = \{3,5,7,8\}$. Also, since "3" is in gap 0 if either state 1 or state 4 prevail, $G_0 = \{1,4\}$.

Similarly, "3" is in gap 1 if either state 2 or 6 prevail, so that $G_1 = \{2,6\}$. Finally, $H_{11} = \{2,5,6,8\}$ indicates the states for which $r_1 = 1$ is in Box 1. Similarly, $H_{12} = \{3,7\}$ indicates that $r_2 = 3$ is in Box 1 in states 3 and 7, $H_{22} = \{5,8\}$ indicates that $r_2 = 3$ is in

Box 2 in states 5 and 8, and $H_{23} = \{6, 7\}$ indicates that $r_3 = 9$ is in Box 2 in states 6 and 7.

4. Solving the Markov Decision Process

The optimal solution to the MDP given in (3) consists of two components: the determination of an *optimal strategy* (or *policy*) that prescribes the action that should be taken for each state of the decision process and the explicit determination of $V(\cdot)$, the optimal value function.

The procedure for determining a solution to the file search problem is done in two parts. In the next subsection, we discuss a procedure, called the Valuation Algorithm, that recursively solves a sequence of $k+1$ finite-stage problems to determine $V(\Pi)$ for any given $\Pi \in \mathcal{S}$. The determination of the optimal strategy forms the second phase of the procedure.

Given the data for a particular problem, it is straightforward to generate all of the possible posterior distributions that could be obtained by examining any combination of boxes. This can be done by first generating all of the possible k-tuples of values that could be observed by any possible search strategy. We refer to these k- tuples as *knowledge states*, since at any stage of the decision process, all that we know for certain about the file is summarized by exactly one of these k-tuples. It is straightforward to write a computer code that generates all possible knowledge states based only on data of the problem. The knowledge states listed in Table 2 were generated by such a routine coded in BASIC.

With the set of knowledge states available, it is then straightforward to generate the set of all possible posterior distributions by repeatedly applying each of the values in each knowledge state to (2). Table 2 below lists all knowledge states and the related posterior distributions for the numerical example. The entries in the columns headed by "Box" indicate the known contents of the relevant box. A "–" indicates that the contents of the box is still unknown. The values in the "Box" columns are generated by the Valuation Algorithm. Notice that there are some knowledge states that are equivalent to one another, in that the associated posterior distributions are identical. The algorithm does not attempt to generate the minimal set of knowledge states, since duplications pose no conceptual problems. Zero probabilities are denoted by blank entries.

	Knowledge State			Posterior Distribution							
Distr.	Box			Core State							
No.	1	2	3	1	2	3	4	5	6	7	8
1	–	–	–	0.504	0.056	0.126	0.216	0.014	0.024	0.054	0.006
2	1	–	–		0.560			0.140	0.240		0.60
3	3	–	–			0.700				0.300	
4	9	–	–				1.000				
5	∅	–	–	1.000							
6	1	3	–					0.700			0.300
7	1	9	–						1.000		
8	1	∅	–		1.000						
9	3	9	–							1.000	
10	3	∅	–			1.000					
11	9	∅	–				1.000				
12	∅	∅	–	1.000							
13	–	3	–					0.700			0.300
14	–	9	–						0.308	0.692	
15	–	∅	–	0.559	0.062	0.140	0.239				

TABLE 2. Knowledge States and Their Associated Posterior Distribution

The procedure for explicitly determining the optimal strategy exploits the fact that the state of the decision process at any stage is the probability distribution over the core states. In the following subsections, we describe the output of the Valuation Algorithm and how it is used to determine an optimal search and disposition strategy.

4.1 Computing $V(\Pi)$: The Valuation Algorithm

The decision process only moves from one stage to the next if a "search" action is taken. In Problem k, there are only k boxes that can possibly be examined. A solution to a $k+1$-stage problem therefore corresponds to the solution to an infinite-horizon problem. (There is not an exogenous requirement that the process stop after a certain number of stages. Since search costs are strictly positive, we know that no optimal strategy will prescribe the continuation of the decision process beyond $k+1$ stages.) For $\Pi \in \mathcal{S}$, let $V_t(\Pi)$ be the optimal expected payoff if the state is Π and the process is forced to stop after t stages if it has not already done so. Therefore, $t = 1, 2, \ldots$ is the maximum number of stages *remaining* in the decision process. The finite-horizon value functions $V_1(\cdot), V_2(\cdot), \ldots$

satisfy the following dynamic programming equations:

$$V_t(\Pi) = \max\Big\{\sum_{i\in S}\pi_i,\ \sum_{i\in G_l}\pi_i,\ l=0,\ldots,k-1,\ J_t(\Pi,j),\ j=1,\ldots,k\Big\}, \qquad (5)$$

where

$$J_t(\Pi,j) = -c + \sum_{q=j}^{n+1} V_{t-1}\left(\Xi_{jq}(\Pi)\right)\cdot\sigma_{jq}(\Pi) \qquad (6)$$

for $t = 1,2,\ldots$ and where $V_0(\Pi)\equiv 0$ for all $\Pi\in\mathcal{S}$.

For a given $\Pi\in\mathcal{S}$, the Valuation Algorithm computes $V(\Pi) = V_{k+1}(\Pi)$, using (5). The algorithm computes $V_{k+1}(\Pi)$ by computing the values of the variables *invalue*, *gapvalue*, and *boxvalue*, which have the following interpretation. The variable *invalue* contains the expected payoff if the r_k is declared to be in the file and the decision process stops. The variable *gapvalue* contains the expected payoff if record r_k is declared to be in the gap whose value is in the variable named *gap*. Finally, *boxvalue* contains the expected payoff if the box whose number is stored in *bestbox* is opened, the distribution is updated according to (2), the decision process moves to the next stage and *proceeds optimally from that point on.* This procedure, coded in BASIC, was used to compute $V_{k+1}(\Pi_1) = 0.89$, where $\Pi_1 = (0.504,\ldots,0.006)$ is the initial distribution for the numerical example. (A psuedocode version of this algorithm is given in the Appendix.) The values of some of the variables determined in the algorithm are: *invalue* = 0.60, *gapvalue* = 0.36, *boxvalue* = 0.89, and *bestbox* = 1. Therefore, it is optimal to open Box 1, examine its contents, update Π_1 to $\Xi_{1q}(\Pi_1)$ if q is observed in Box 1, move to stage 2 and proceed from there on optimally.

What action should we take at stage 2? The action we take at stage 2 depends upon what we observed in Box 1. Suppose, for example, that we observe a "1" in Box 1. Then, using (2), $\Pi_2 = (0, 0.560, 0, 0, 0.140, 0.240, 0, 0.60)$. We can use the Valuation Algorithm to determine $V_{k+1}(\Pi_2)$ and the optimal action to take if the initial distribution is Π_2. The optimality of this action, however, depends only on the state and not on the stage of the decision process. Therefore, the *action* prescribed by the algorithm when Π_2 is the state is the optimal action to take after a "1" is observed in Box 1.

To determine $V_{k+1}(\Pi_2)$, the algorithm specifies the following values: *invalue* = 0.60, *gapvalue* = 0.4, *boxvalue* = 0.9, and *bestbox* = 2. It is clear that $V_{k+1}(\Pi_2) = 0.9$ and it is now optimal to pay $c = 0.1$ and open Box 2! The fact that we just spent $c = 0.1$ to examine Box 1 is no longer relevant. Since it is optimal to examine Box 2 and we are looking for

"3", the second smallest record value, the remaining contingencies are trivial–we will know for certain if "3" is in file or, if it isn't, what the gap will be. We exploit the fact that the Valuation Algorithm can be used to determine the optimal action to take as a function of *any* distribution on $\mathcal{C}$. The fact that we are computing the optimal expected payoff for an infinite-horizon problem when in fact we are in some stage t is not relevant. We compute $V_{k+1}(\cdot)$ when the decision process is actually in stages less than $k+1$, not to determine the absolute magnitude of the payoffs associated with each of the feasible actions, but to see which action yields the highest payoff given the information available at that stage.

4.2 An Optimal Search and Disposition Strategy

We identify an optimal search and disposition strategy in the following way. For each distribution in Table 2, compute $V_{k+1}(\cdot)$ and the value of relevant variables using the Valuation Algorithm. Table 3 displays the information generated by the algorithm for each of the 15 distributions in Table 2. The initial distribution is in Row 1. (Notice that the knowledge state in Row 1 indicates that none of the boxes have been examined.) The value of $V_{k+1}(\cdot)$ is 0.89, the optimal expected payoff the decision process. The optimal action is to examine Box 1 since $boxvalue = 0.89$ and $bestbox = 1$. Suppose we observe a "1" in Box 1. The knowledge state is, therefore, (1,–,–), which is Row 2 of Table 2. The posterior distribution associated with this knowledge state is in Row 2 of Table 2. The action that yields the highest expected payoff, given we begin the decision process in this knowledge state, is to examine Box 2. (The value of $boxvalue = V_1(\cdot)$ and $bestbox = 2$.) Suppose that we observe a "9" in Box 2. The new knowledge state is (1,9,–), which is Row 7 of Table 3. We see, of course that the optimal action to take in this knowledge state is stop the process and declare that "3" is in gap 1. This is the strategy discussed earlier.

Since every possible knowledge state is listed in Table 3, this table summarizes the complete contingency plan the comprises the optimal search and disposition strategy. Notice that in an alternate (finite-state) formulation of the problem that has as its states the values known to be in the each of the boxes at any stage of the decision process would constitute a table much like Table 2. In this sense, the effort required to determine the values in Table 2 cannot be avoided. The procedure here computes this table and then evaluates each of the possible states.

	Knowledge State			Payoff Values					
Distr.	Box								
No.	1	2	3	*invalue*	*gapvalue*	*gap*	*boxvalue*	*bestbox*	$V(\cdot)$
1	–	–	–	0.600	0.360	0	0.89	1	0.890
2	1	–	–	0.600	0.400	1	0.90	2	0.900
3	3	–	–	1.000	0.000	0	0.90	1	1.000
4	9	–	–	0.000	1.000	0	0.90	1	1.000
5	$\emptyset$	–	–	0.000	1.000	0	0.90	1	1.000
6	1	3	–	1.000	0.000	0	0.90	1	1.000
7	1	9	–	0.000	1.000	1	0.90	1	1.000
8	1	$\emptyset$	–	0.000	1.000	1	0.90	1	1.000
9	3	9	–	1.000	0.000	0	0.90	1	1.000
10	3	$\emptyset$	–	1.000	0.000	0	0.90	1	1.000
11	9	$\emptyset$	–	0.000	1.000	0	0.90	1	1.000
12	$\emptyset$	$\emptyset$	–	0.000	1.000	0	0.90	1	1.000
13	–	3	–	1.000	0.000	0	0.90	1	1.000
14	–	9	–	0.931	0.069	1	0.90	1	0.931
15	–	$\emptyset$	–	0.551	0.408	0	0.90	1	0.900

TABLE 3. Evaluation of Posterior Distributions

There are several interesting features of the computational scheme proposed here. The formulation given in (3) is a POMDP, whose state space is the continuum $\mathcal{S}$. While there are methods for solving infinite-horizon POMDP's (see Monahan (1982) and references therein), they are not nearly as efficient as algorithms for solving finite-state and action MDP's. The procedure suggested here effectively reduces the computational burden required to solve this particular POMDP to that of a finite-state, finite-action, and finite-horizon MDP. The optimal solution to such an MDP is easily determined by generating tables of payoffs and actions yielding those payoffs for all stages in the planning horizon. The optimal strategy is then determined by the values in the tables. See Hillier and Lieberman (1985) for an example of such a procedure. The method used here is a variation of this backward substitution method. The sequence of activities leading to the optimal strategy, however, differs from conventional methods. In spite of this difference, the overall computational effort of the two procedures are equivalent.

5. Analytical Results

The formulation of the problem with the state of the decision process being a distribution over core states is particularly amenable to sensitivity analysis that determines how changes in parameters of the problem influence both the optimal expected payoff and an optimal search and disposition strategy. In this section, we establish several properties of these functions. Several of the results are established by showing that they hold for the finite-horizon version of the problem given in (5) and (6) and that these properties persist as the planning horizon goes to infinity. (In fact we know that for any $t > k$, $V(\Pi) = V_t(\Pi)$ for any $\Pi \in \mathcal{S}$.) Therefore, properties established for $V_t(\cdot)$ for every t also hold for $V(\cdot)$.

5.1 Qualitative Results for the General Problem

The first result is used to characterize the set of states at which it is optimal to stop or to search.

PROPOSITION 1. *$V_t(\Pi)$ is convex on $\mathcal{S}$ for $t = 0, 1, \ldots$.*

PROOF: The result is easily established by induction on t. The most difficult step is to show that $J_t(\Pi, j)$ is convex in Π if $V_{t-1}(\Pi)$ is convex. This result is well-known, however. See, for example, Astrom (1969, Lemma 2) or DeGroot (1970, Lemma 1, page 435).

The convexity of the optimal value functions and the linearity of the payoffs when a stop action is taken leads immediately to the convexity of the sets of states at which it is optimal to stop and declare either that r_k is in the file or that r_k is in some gap. Let

$$\mathcal{S}_{G_l} = \{\Pi \in \mathcal{S} \mid V(\Pi) = \sum\nolimits_{i \in G_l} \pi_i\}$$

be the set of states for which stopping and declaring the record to be in gap l, $l = 0, \ldots, k-1$ is optimal. Let

$$\mathcal{S}_I = \{\Pi \in \mathcal{S} \mid V(\Pi) = \sum\nolimits_{i \in S} \pi_i\}$$

be the set of states such that stopping and declaring the record to be in the file is optimal.

COROLLARY 1. *The sets $\mathcal{S}_{G_l}$, $l = 0, \ldots, k-1$, and $\mathcal{S}_I$ are convex.*

PROOF: We prove that $\mathcal{S}_I$ is convex. The demonstration that $\mathcal{S}_{G_l}$ is convex is analogous. For $\Pi_1, \Pi_2 \in \mathcal{S}_I$ and $0 \leq \lambda \leq 1$, let $\widehat{\Pi} = (\hat{\pi}_1, \ldots, \hat{\pi}_{2^n}) = \lambda\Pi_1 + (1-\lambda)\Pi_2$. Then

$V(\widehat{\Pi}) \leq \lambda V(\Pi_1)+(1-\lambda)V(\Pi_2) = \sum_{i\in S}\hat{\pi}_i$, where the inequality follows from the convexity of $V(\cdot)$. But $V(\widehat{\Pi}) \geq \sum_{i\in S}\hat{\pi}_i$, so equality must hold and $\widehat{\Pi} \in \mathcal{S}_I$.

Let $V(\Pi, c)$ explicitly reflect the dependence of optimal expected payoffs on the search cost. The next result establishes the intuitive characteristic that optimal payoffs decline as the search cost increases and furthermore that optimal marginal payoffs also decline.

PROPOSITION 2. *For $\Pi \in \mathcal{S}$, $V_t(\Pi, c)$ is nonincreasing and convex in c for $t = 0, 1, \ldots$.*

PROOF: Again, the proof is by induction on t. Since $V_1(\Pi, c)$ is independent of c, the result is vacuously true. Assume that $V_{t-1}(\Pi, c)$ is decreasing and convex in c. Then $J_t(\Pi, j, c) = -c + \sum_{q=j}^{n+1} V_{t-1}(\Xi_{jq}(\Pi), c) \cdot \sigma_{jq}(\Pi)$ is also convex and decreasing in c, since it is the sum of a convex function of c and the weighted average of convex functions of c. Since the maximum of convex, decreasing functions inherits these properties, the proof is complete.

The characterization of the optimal payoffs as a function of the search cost leads immediately to another intuitive result.

COROLLARY 2. *There exits a $c^* \in (0, 1)$, such that for $c \geq c^*$ it is never optimal to search any box.*

We now establish the fact that having perfect information about the contents of a box cannot diminish optimal expected payoffs. This result, while interesting in its own right, is also used in the next subsection to characterize the optimal search strategy for a special case. For convenience, several intermediate definitions and results are required. Let $\alpha(\Pi)$ be the expected payoff if a stop action is taken when the state is $\Pi \in \mathcal{S}$; i.e.,

$$\alpha(\Pi) = \max\Big\{\sum\nolimits_{i\in S}\pi_i,\ \sum\nolimits_{i\in G_l}\pi_i,\ \text{for } l = 0, \ldots, k-1\Big\}. \tag{7}$$

The next results describe how payoffs associated with stopping depend upon the stage of the decision process. The first result is standard in the analysis of MDP models and is stated without proof: as the planning horizon lengthens, optimal payoffs cannot diminish. The second part establishes that if it is optimal stop when at most t stages remain, it is optimal to stop if at most $t-1$ stages remain.

LEMMA 1. *For $\Pi \in \mathcal{S}$ and $t = 2, 3, \ldots$,*

a. $V_t(\Pi) \geq V_{t-1}(\Pi)$

b. *If $V_t(\Pi) = \alpha(\Pi)$, then $V_{t-1}(\Pi) = \alpha(\Pi)$.*

PROOF: (Part b.) Suppose (b) is false. Then $V_{t-1}(\Pi) > \alpha(\Pi) = V_t(\Pi)$, which contradicts Lemma 1-a.

We now examine how optimal payoffs are influenced by some of the initial probabilities that records are in the file. In particular, we establish the intuitively appealing results that in Problem k the expected payoff associated with declaring r_k to be in the file is nondecreasing in p_k and that the payoff associated with declaring r_k to be in gap l, $l = 0, \ldots, k-1$, is nonincreasing in p_k. Before doing this, however, we establish some preliminary results. Let $\Pi(p_k)$ denote the initial state of the decision process when p_k is the initial probability that r_k is in the file. Let $\pi_i(p_k)$ be the ith component of $\Pi(p_k)$. Analogously, $\Pi_t(p_k)$ denotes the distribution at the beginning of period t.

The next results are used to establish the dependence of any posterior probability on p_k. For notational convenience, let $S^c \equiv \{i \mid i \in \mathcal{C}, i \in G_l, \text{ for some } l = 0, 1, \ldots, k-1\}$ denote the complement of S.

LEMMA 2. *For $t \geq 2$,*

$$\pi_{ti} = \begin{cases} 0 & \text{if } \pi_{t-1,i} = 0 \\ \dfrac{\pi_i}{\sum_{m \in I} \pi_m} & \text{if } \pi_{t-1,i} > 0, \text{ where } I \subset \mathcal{C}. \end{cases}$$

PROOF: The proof is by induction on t. Using (1) and (2), the result holds at $t = 2$. For Π_{t-1}, we observe the occurrence of the event $I \subset \mathcal{C}$ and this knowledge is used to construct Π_t. Assume that the result holds for some $t \geq 2$ so that

$$\pi_{ti} = \begin{cases} \dfrac{\pi_i}{\sum_{m \in I} \pi_m} & \text{if } \pi_{t-1,i} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

where $I \subset \mathcal{C}$. Suppose we examine Box j and observe r_q. If $\pi_{ti} = 0$, then $\Xi_{jqi}(\Pi_t) = 0$.

Suppose that $\pi_{ti} > 0$. Then, from (1) and (2),

$$\begin{aligned}\Xi_{jqi}(\Pi_t) &= \frac{\pi_{ti}}{\sum_{l\in H_{jq}} \pi_{tl}} \\ &= \frac{\frac{\pi_i}{\sum_{m\in I}\pi_m}}{\sum_{l\in H_{jq}} \frac{\pi_l}{\sum_{m\in I}\pi_m}} \\ &= \frac{\pi_i}{\sum_{l\in H_{jq}} \pi_l},\end{aligned}$$

and the result holds for all t.

LEMMA 3.

a. *For* $i \in S$, $\pi_i(p_k) = p_k \frac{\partial \pi_i}{\partial p_k}$.

b. *For* $i \in S^c$, $\pi_i(p_k) = -(1-p_k)\frac{\partial \pi_i}{\partial p_k}$.

PROOF: For any $i \in S$, $\pi_i = \kappa_i p_k$, where $\kappa_i \geq 0$ is the product of some combination of p_j or $(1-p_j)$ for all $j \neq k$ and does not depend upon p_k. Therefore

$$\frac{\partial \pi_i}{\partial p_k} = \kappa_i = \frac{\pi_i(p_k)}{p_k},$$

and (a) is established.

Similarly, for $i \in S^c$,

$$\frac{\partial \pi_i}{\partial p_k} = -\kappa_i = \frac{-\pi_i(p_k)}{1-p_k},$$

so that part (b) also holds.

Using these facts, we can now establish the influence of p_k on the state distribution.

PROPOSITION 3. *For* $t = 1, 2, \ldots,$

$$\pi_{ti}(p_k) \text{ is } \begin{cases} \text{nondecreasing in } p_k \text{ if } i \in S \\ \text{nonincreasing in } p_k \text{ if } i \in S^c. \end{cases}$$

PROOF: The proof is by induction on t. For $t = 1$, the result follows immediately from Lemma 3 since $\kappa_i > 0$ for all i. Assume the result holds for some $t \geq 2$.

Suppose that an examination of Box j reveals record r_q. Then, using Lemma 2, the ith component of $\Xi_{jq}(\Pi_t(p_k))$ is

$$\Xi_{jqi}(\Pi_t(p_t)) = \frac{\pi_i(p_k)}{\sum_{m\in H_{jq}} \pi_m(p_k)}.$$

if $\pi_{ti}(p_t) > 0$. Therefore, for $i \in S$ such that $\pi_{ti}(p_t) > 0$,

$$\begin{aligned}
\frac{d\Xi_{jqi}(\Pi_t(p_t))}{dp_t} &= \sigma_{jq}(\Pi_t(p_k))\frac{\partial \pi_i}{\partial p_k} - \pi_i \sum_{m \in H_{jq}} \frac{\partial \pi_m}{\partial p_k} \\
&= \left[\sum_{l \in H_{jq} \cap S} p_k \frac{\partial \pi_l}{\partial p_k} - \sum_{l \in H_{jq} \cap S^c} (1 - p_k)\frac{\partial \pi_l}{\partial p_k} \right] \frac{\partial \pi_i}{\partial p_k} - \left(p_k \frac{\partial \pi_i}{\partial p_k} \right) \sum_{m \in H_{jq}} \frac{\partial \pi_m}{\partial p_k} \\
&= \left(\sum_{m \in H_{jq}} \frac{\partial \pi_m}{\partial p_k} \right) \frac{\partial \pi_i}{\partial p_k} p_k[1 - 1] - \frac{\partial \pi_i}{\partial p_k} \sum_{l \in H_{jq} \cap S^c} \frac{\partial \pi_l}{\partial p_k} > 0,
\end{aligned}$$

where the second equality follows from Lemma 3. The inequality follows since $\frac{\partial \pi_i}{\partial p_k} > 0$ from the induction hypothesis and $\frac{\partial \pi_l}{\partial p_k} < 0$ for all $l \in H_{jq} \cap S^c$.

For $i \in S^c$ such that $\pi_{ti}(p_k) > 0$,

$$\begin{aligned}
\frac{d\Xi_{jqi}(\Pi_t(p_t))}{dp_t} &= \sigma_{jq}(\Pi_t(p_k))\frac{\partial \pi_i}{\partial p_k} - \pi_i \sum_{m \in H_{jq}} \frac{\partial \pi_m}{\partial p_k} \\
&= \left[\sum_{l \in H_{jq} \cap S} p_k \frac{\partial \pi_l}{\partial p_k} - \sum_{l \in H_{jq} \cap S^c} (1 - p_k)\frac{\partial \pi_l}{\partial p_k} \right] \frac{\partial \pi_i}{\partial p_k} + (1 - p_k)\frac{\partial \pi_i}{\partial p_k} \sum_{m \in H_{jq}} \frac{\partial \pi_m}{\partial p_k} \\
&= \frac{\partial \pi_i}{\partial p_k} \sum_{l \in H_{jq} \cap S} \frac{\partial \pi_l}{\partial p_k} < 0
\end{aligned}$$

where the inequality follows from the induction hypothesis that $\frac{\partial \pi_i}{\partial p_k} < 0 (> 0)$ for $i \in S^c (\in S)$, respectively. The result therefore holds for all t.

Since the expected payoffs associated with stopping are sums of $\pi_{ti}(p_k)$, it follows immediately that these payoffs are also monotonic in p_k.

COROLLARY 3.

i. $\sum_{i \in S} \pi_i(p_k)$ *is nondecreasing in* p_k *and*

ii. $\sum_{i \in G_l} \pi_i(p_k)$ *is nonincreasing in* p_k *for* $l = 0, \ldots, k-1$.

We conclude the discussion of the characteristics of optimal search and disposition strategies by examining the special case when each of the records are equally likely to be included in the file and when that common probability is 0.5.

5.2 Special Case: Problem $n = 3$ when $p_i = 0.5$ for all i.

We now examine the example in Section 3.3 when $k = 3$ and $p_i = 0.5$ for all i. In some sense, this is the "maximum uncertainty" case–all records are equally likely to be included and, since this is Problem 3, all boxes are potential candidates for search. We use this example to illustrate several interesting properties of optimal search and disposition strategies.

Table 4 below shows expected payoffs and optimal actions for several knowledge states (those that are associated with uncertainty regarding the status of $r_3 = 9$). The cost of searching one box is $c = 0.28$. The column headed by *Box i* contains the expected payoff if Box i is examined first and an optimal policy is followed thereafter. The "Optimal Action" column indicates the optimal first action as a function of the state of the decision process.

	Knowledge State			Payoff Values						
Distr.	Box									Optimal
No.	1	2	3	*invalue*	*gapvalue*	*gap*	*Box*1	*Box*2	*Box*3	Action
1	–	–	–	0.50	0.25	1	0.44	0.51	0.34	Box 2
2	1	–	–	0.50	0.25	1	0.30	0.58	0.51	Box 2
3	3	–	–	0.50	0.50	1	0.44	0.72	0.44	Box 2
6	1	3	–	0.50	0.50	2	0.44	.044	0.72	Box 3
22	1	–	$\emptyset$	0.33	0.33	1	0.44	0.72	0.44	Box 2
23	3	–	9	0.50	0.50	1	0.44	0.72	0.44	Box 2
24	3	–	$\emptyset$	0.50	0.50	1	0.44	0.72	0.44	Box 2
27	–	3	–	0.50	0.50	2	0.44	0.44	0.72	Box 3
29	–	$\emptyset$	–	0.25	0.50	1	0.72	0.44	0.44	Box 1
33	–	$\emptyset$	$\emptyset$	0.25	0.50	1	0.72	0.44	0.44	Box 1
35	–	–	$\emptyset$	0.43	0.29	1	0.52	0.56	0.28	Box 2

TABLE 4. Evaluation of Some Posterior Distributions

For convenience, Π_i will refer to the distribution number i in Table 4. There are several observations that can be made from the data in Table 4:

1. $\alpha(\Pi_1) = 0.5 > \alpha(\Pi_{35}) = 0.43$, where $\Pi_{35} = \Xi_{3\emptyset}(\Pi_1)$ (i.e., Π_{35} is the posterior distribution when Π_1 is the prior, Box 3 is opened and is found to be empty). Therefore,

$$\alpha(\Xi_{jq}(\Pi)) \not> \alpha(\Pi), \tag{8}$$

for j, q.

2. Relation (8) illustrates that in general $V(\Xi_{jq}(\Pi)) \not> V(\Pi)$ for all j and q. In particular, note that for c large, testing is never optimal for any Π and $V(\Pi) = \alpha(\Pi)$.

3. When there is no information regarding the contents of the file (i.e., Π_1 is the current distribution), it is optimal to examine the median box first (Box 2). A more detailed discussion of strategies of this form follows.

4. If there is ever uncertainty regarding the status of $r_3 = 9$, it is optimal search.

Since at most three boxes need to be examined, the evaluation of the explicit optimal payoff as a function of the search cost is straightforward. The fact that $p_i = \frac{1}{2}$ also simplifies the computation of the revised probabilities resulting from the acquisition of information. Let $B_i(c)$ be the expected payoff when Box i is searched first and an optimal policy is followed thereafter, given that c is the search cost. Expressions for $B_i(c)$ follow when the initial state of the decision process is Π_1:

Look in Box 2 first:

$$B_2(c) = \begin{cases} -c \\ +\frac{2}{8}\max\begin{cases} \frac{1}{2}(1) & \text{Declare "In"} \\ \frac{1}{2}(1) & \text{Declare "Gap 2"} \\ -c+1 & \text{Examine Box 3} \end{cases} & \text{See "3" in Box 2} \\ +\frac{2}{8}(1) & \text{See "9" in Box 2} \\ +\frac{1}{2}\max\begin{cases} \frac{1}{4}(1) & \text{"In"} \\ \frac{1}{4}(1) & \text{"Gap 0"} \\ \frac{1}{2}(1) & \text{"Gap 1"} \\ -c+1 & \text{Examine Box 1} \end{cases} & \text{See "}\emptyset\text{" in Box 2} \end{cases}$$

Look in Box 1 First:

$$B_1(c) = \left\{ \begin{array}{l} -c \\ +\frac{1}{8}(1) \quad \text{See “}\emptyset\text{”} \\ +\frac{1}{2}\max\left\{ \begin{array}{ll} \frac{1}{2} & \text{Declare “In”} \\ \frac{1}{4} & \text{Declare “Gap 1”} \\ \frac{1}{4} & \text{Declare “Gap 2”} \\ -c+\frac{1}{4}(1) & \text{See “9”} \\ & +\frac{1}{2}\max\left\{ \begin{array}{lll} \frac{1}{2} & \text{“In”} & \\ \frac{1}{2} & \text{“Gap 2”} & \text{Look Box 2} \\ -c+1 & \text{Look Box 3} & \end{array} \right. \\ & +\frac{1}{4}(1) \quad \text{See “99”} \\ -c+\frac{1}{4}(1) & \text{See “9”} \\ & +\frac{3}{4}\max\left\{ \begin{array}{ll} \frac{1}{3} & \text{“In”} \\ \frac{1}{3} & \text{“Gap 0”} \quad \text{Look Box 3} \\ \frac{1}{3} & \text{“Gap 1”} \\ -c+1 & \text{Look Box 2} \end{array} \right. \end{array} \right. \text{See “1”} \\ +\frac{2}{8}\max\left\{ \begin{array}{lll} \frac{1}{2} & \text{“In”} & \\ \frac{1}{2} & \text{“Gap 1”} & \text{See “3”} \\ -c+1 & \text{Look Box 2} & \end{array} \right. \\ +\frac{1}{8}(1) \quad \text{See “9”} \end{array} \right.$$

Look in Box 3 First:

$$
B_3(c) = \begin{cases} -c \\ \quad + \frac{7}{8} \max \begin{cases} \frac{3}{7}(1) & \text{Declare "In"} \\ \frac{1}{7}(1) & \text{Declare "Gap 0"} \\ \frac{1}{7}(1) & \text{Declare "Gap 1"} \\ \frac{1}{7}(1) & \text{Declare "Gap 2"} \\ -c + \frac{2}{7}(1) + \frac{1}{7}(1) + \frac{4}{7} \max \begin{cases} \frac{1}{4} & \text{"In"} \\ \frac{1}{2} & \text{"Gap 1"} \\ \frac{1}{4} & \text{"Gap 0"} \\ -c+1 & \text{Look Box 1} \end{cases} & \text{Look Box 2} \\ -c + \frac{1}{7}(1) + \frac{2}{7} \max \begin{cases} \frac{1}{2} & \text{"In"} \\ \frac{1}{2} & \text{"Gap 1"} \\ -c+1 & \text{Look Box 2} \end{cases} & \text{See "99" in Box 3} \\ \quad + \frac{3}{7} \max \begin{cases} \frac{1}{3} & \text{"In"} \\ \frac{1}{3} & \text{"Gap 1"} \\ \frac{1}{3} & \text{"Gap 2"} \\ -c+1 & \text{Look Box 2} \end{cases} & \text{Look Box 1} \\ \quad + \frac{1}{7}(1) \end{cases} \\ \quad + \frac{1}{8}(1) \quad \text{See "9" in Box 3} \end{cases}
$$

The dynamic program is now written in terms of the $B_i(c)$ functions:

$$
V(\Pi_1) = \begin{cases} \frac{1}{8}(4) & \text{Declare "9" to be in the file} \\ \frac{1}{8} & \text{Declare Gap "0"} \\ \frac{1}{8}(2) & \text{Declare Gap "1"} \\ \frac{1}{8} & \text{Declare Gap "2"} \\ B_1(c) & \text{Look in Box 1 first} \\ B_2(c) & \text{Look in Box 2 first} \\ B_3(c) & \text{Look in Box 3 first} \end{cases} \tag{9}
$$

Note that if $c < \frac{1}{2}$, then $-c+1 > \frac{1}{2} > \frac{3}{7} > \frac{1}{3}$ so that it is always optimal to continue to search if a search action has ever been taken. (In each of the "max" functions inside $B_i(c)$, $c < \frac{1}{2}$ implies that the expected payoff associated with searching exceeds the expected payoff when a stop action is taken.) Under this condition,

$$
\begin{aligned} B_1(c) &= -2c + 1 \\ B_2(c) &= -\frac{7}{4}c + 1 \\ B_3(c) &= -\frac{19}{8}c + 1. \end{aligned}
$$

We see, therefore, that $B_2(c) \geq B_1(c) \geq B_3(c)$ for all $c < \frac{1}{2}$. If it is optimal to look in a box, it is optimal to examine the second box first and to continue to search until the status of $r_3 = 9$ is known with certainty. Using (9), it is optimal to initiate search if, and only if,

$$B_2(c) > \frac{1}{2} \Leftrightarrow c < \frac{2}{7}.$$

In the example in Table 4, $c = .28 < \frac{2}{7}$, so the properties discussed in points (3) and (4) are special cases of these observations.

The optimal search and disposition strategy for Problem 3 with $p_i = \frac{1}{2}$ for all i is summarized as follows: If $c < \frac{2}{7}$, search, beginning with Box 2, until the status of $r_3 = 9$ is known for certain; otherwise, stop and declare that $r_3 = 9$ is in the file.

When $c < \frac{2}{7}$, seach will always occur if there is some uncertainty regarding the status of the record. It is easy to show via numerical example that such a "search-while-uncertainty-persists" stategy is typically not optimal for arbitrary values of p_i. When $n = 3$, $p_1 = 0.1$, $p_2 = 0.3$, $p_3 = 0.8$, and $c = 0.31$, and the problem is to determine the status of r_3, the optimal strategy is to examine Box 1 initially. If r_1 is observed in Box 1, it is optimal to stop and declare that r_3 is in Gap 1 even though uncertainty regarding the status of r_3 persists.

The optimal search pattern in the "search-while-uncertainty-exits" example is that of a "balanced" search tree discussed in Knuth (1973): if any search is optimal, it is optimal to examine the median box first and eliminate half of the file. If the object is not found, continue examining the median box of those boxes that have not yet been eliminated.

It a clear that searching the median box depends crucially on the assumption that $p_i = 0.5$. Consider the following variation of the example: Again, the objective is to determine the status of $r_3 = 9$. Assume now that $p_1 = p_2 = p_3 = p$, for some $0 \leq p \leq 1$, so that it is equally likely for each of the three records to be in the file. Suppose that $c = 0.01$, a relatively low value so that it is economical to examine at least one of the boxes. (Corollary 2 tells us that there could be search costs that preclude search.) Which box should be searched first? From the discussion above, we conjecture that a binary search strategy that examines the middle of the file—look in Box 2 in this case—is optimal if $p = 0.5$. This strategy is not optimal for all values of p, however. When p is near one, for example, it is clear that if a box is to be examined, it should be Box 3. (Numerical calculations show that for any $p > 0.8$, it is optimal to examine Box 3 first.) On the other

hand, when p is near 0, numerical calculations show that it is optimal to examine Box 1 first. Here is where the ramification of the sequential storage requirement is most evident. When p is near 0, it likely that there are few records in the file. An examination of the first box is therefore the source of a significant amount of information. Finally, when p takes on intermediate values, numerical calculations show that it is indeed optimal to examine Box 2, the median box, first. We see that even in the relatively simple case where the p_i's are equal, it can be difficult to predicate where to begin the search. The problem becomes even more complex when the p_i's differ across records.

6. Summary

A problem of search for a hidden object when an ordering relation holds was formulated as an infinite-horizon Markov decision process whose state space is a continuum, consisting of the space of probability distributions over a finite set of core states. A variation of a procedure for computing solutions to MDP's with a finite number of states, actions, and stages was used to compute optimal expected payoffs and an optimal search and disposition strategy. The model formulation made it possible to derive several qualitative characteristics of the optimal expected payoff function as well as the optimal search and disposition strategy.

Acknowledgement

This research was completed while the author was on sabbatical leave at the London School of Economics and Political Science. He would like to thank the members of the Operational Research group in the Department of Statistical and Mathematical Sciences at LSE for their support. The author would like to thank his colleague Susan Cohen for many helpful discussions regarding information and its acquisition and an extremely diligent referee for pointing out a number of technical problems in earlier versions of the paper. Their comments and criticisms are greatly appreciated.

Appendix

Valuation Algorithm

$invalue := \sum_{i \in S} \pi_i$;
$gap := n$; $gapvalue = -1000$; 'Initial Values
for $l := 0$ **step** 1 **until** $k - 1$ **do**
  $x := \sum_{i \in G_l} \pi_i$;
  if $x > gapvalue$ **then**
    $gapvalue := x$; $gap := l$;
  next l;
if $t = k$ **then** $boxvalue := -c$;
  else
    $bestbox := 0$; $boxvalue := -1000$;
    for $j := 1$ **step** 1 **until** k **do**
      $sum := -c$;
      for $q := j$ **step** 1 **until** $n + 1$ **do**
        'q indexes records in Box j
        Compute $\Xi_{jq}(\Pi)$ **using** (2);
        $sum := V_{t+1}(\Xi_{jq}(\Pi)) \cdot \sigma_{jq}(\Pi) + sum$;
        next q;
      if $sum > boxvalue$ **then**
        $boxvalue = sum$; $bestbox = j$;
      next j;
$V_t(\Pi) := \max\{invalue, gapvalue, boxvalue\}$;
end;

References

Astrom, K. J., "Optimal Control of Markov Processes with Incomplete State-Information II. The Convexity of the Lossfunction," *J. Math. Analysis*, 26 (1969), 403–406.

Blair, C. and G. E. Monahan, "A Dynamic Programming Approach to Finding Records in a File," Unpublished manuscript, Department of Business Administration, University of Illinois, Champaign, IL, January 1990.

De Groot, M. H., *Optimal Statistical Decisions*, McMillan, New York, NY, 1970.

Hillier, F. S. and G. L. Lieberman, *Operations Research, 3rd Ed.*, Holden-Day, San Francisco, CA, 1985.

Knuth, D. E., "Optimal Binary Search Trees," *Acta Informatica*, 1 (1971),14–25.

————, "Binary Searching Trees," Chapter 6 in *Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

Monahan, G. E., "A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms," *Management Science*, 28 (Jan. 1982), 1–16.

Moore, J. C., W. B. Richmond, and A. B. Whinston, "A Decision Theoretic Approach to File Search," *Computer Science in Economics and Management* 1 (1988), 3–19.

———— and A. B. Whinston, "A Model of Decision-Making with Sequential Information-Acquisition (Part I)," *Decision Support Systems*, 2 (1986), 285–307.

———— and ————, "A Model of Decision-Making with Sequential Information-Acquisition (Part II)," *Decision Support Systems*, 3 (1987), 47–72.

Stone, L., *Theory of Optimal Search*, Academic Press, New York, NY, 1975.

Wiederhold, G., "The Sequential File," Section 3-2 in *Database Design*, McGraw-Hill Book Company, New York, NY, 1977, 86–94.

Weitzman, M, "Optimal Search for the Best Alternative", *Econometrica*, 47 (May 1979), 637–654.