SEARCH

J.H. O'Geran, *City University, London.*
H.P. Wynn, *City University, London.*
A.A. Zhigljavsky, *Leningrad University, St. Petersburg.*

Keywords: Search,screening,optimization,entropy,sequential design, Bayes.

Abstract. The paper reviews the theory of search. The main viewpoint is statistical but considerable space is spent on connections with other fields such as logic and optimization. Three primitive ideas dominate: consistency, entropy and Bayes methods. The theory of screening provides one of the central examples and there is a section on non-sequential random search deriving from the work of Renyi. The game of "bulls and cows" provides a motivating example for both sequential and non-sequential algorithms.

1. Introduction

Search is a vast area if a simple paper-count is made. It also straddles many different disciplines. Because of the normal intellectual separation between disciplines, it is not surprising to find the same ideas rediscovered or dressed differently in various places. Fortunately, however, the differences seem to be crumbling and we have been able to make a synthesis. We have conciously given the paper a statistical slant and tried to draw together some of the results in the statistical literature while making connections across to other disciplines. Within statistics screening and group-testing are areas where search methodologies have been employed directly but otherwise search has been a subliminal and even neglected subject.

There is a view that search may be a fundamental component to epistomology. There is not the space to develop this idea here but it is enlightening to quote from the American pragmatist philosopher C.S. Pierce on the game of 20 Questions (Pierce, 1901)

"...twenty skillfil hypotheses will ascertain what two hundred thousand stupid ones might fail to do. The secret of the business lies in the caution which breaks a hypothesis into its smallest logical components, and only risks one of them at a time. What a world of futile controversy might have been saved if this principle had guided investigations..."

The idea of bifurcation, inherent in the game of questions is, of course, an ingredient of many search methods, logic (true, false) and binary coding theory. The links

into these subjects are spelled out here but some of the more profound relationships such as between search, packing information and computability are only touched on. These are topics of research by the authors and they would be happy to stimulate others to work on the connections. It is useful to cast search in the framework of "observer logics" in which the result of observation is to impose futher constraints or axioms on the system. If the authors were forced to choose then this would probably be the preferred formulation.We note that we have not devoted space to globally optimal dynamic programming (Bellman) solutions which prove to be intractable for many non-linear search problems of the kind we cover.

A broad separation in the field is between sequential and non-sequential procedures. Although most of the review is taken up by sequential procedures we use the opportunity to expose some basic work on non-sequential procedures which can provide bench-marks for the optimality of sequential methods. Understanding the distinction between the two is important.

2.A set formulation

Search is an active process. It has a similar status to sequential experimentation, dynamic control, optimization and a host of areas in which decisions need to be made before and after observation. It could be cast as a "dynamic decision process" or in some similar framework but we wish to avoid particular subject-labels. The primitive notion adopted here is that the searcher selects a *test set* X to try to identify a *target set* T. If X is chosen in the "right place" then it should interact with T in some sence. For example if T and X have some elements in common, or otherwise, then this may be recorded in some way.

Proceeding formally let $\mathbf{X}$ be a collection of *units*. The units may be subjects, parameters, items, components etc. The target field $\mathcal{T}$ is a member of $\mathcal{B}(\mathbf{X})$ the Borel subsets of $\mathbf{X}$. Thus $\mathcal{B}(\mathbf{X})$ includes all finite intersections and subsets of $\mathbf{X}$. In much of what follows $\mathbf{X}$ will be finite. Write in this case

$$\mathcal{T} = \{T_1, T_2, ...\} \subseteq \mathcal{B}(\mathbf{X})$$

so that the target $T = T_j$ for some j. A simplification in this review, which will be retained throughout except where explicitly changed, is to take the search field also to be subsets of the same units:

$$\mathcal{X} \subseteq \mathcal{B}(\mathbf{X})$$

In very special cases $\mathcal{T} = \mathcal{X}$ but such cases are of considerable interest (see Section 11 on the game of "bulls and cows"). The interaction between $X \in \mathcal{X}$ and $T \in \mathcal{T}$ is measured by a non-negative *search function*

$$f : \mathcal{X} \times \mathcal{T} \rightarrow \mathbf{R}_+.$$

an evaluation of this function in a real situation is thus written $f(X,T)$ and is referred to as an *observation*. A *search problem* is the quadruple

$$\mathcal{J} \;=\; (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$$

Sometimes we omit $\mathbf{X}$ from the quadruple.

The first primitive notion concerns the identifiability of the search problem.

Definition 1. A target $t \in \mathcal{T}$ is said to be achievable within the search problem $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ if for any $T' \in \mathcal{T}$ such that

$$f(X,T) = f(X,T') \;\text{ for all } X \in \mathcal{X}$$

it follows that $T = T'$.

The idea here is that the search field together with the search function f is enough to *distinguish* the target T. Now since T will typically be unknown and may potentially take any "value" in $\mathcal{T}$, the last definition leads naturally to

Definition 2. A search problem $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ is *solveable* if every T in $\mathcal{T}$ is achievable within $\mathcal{J}$.

Solving a search problem $\mathcal{J}$ means finding the target set T or getting "close" in some sense. Naturally observing f at every set X is usually not efficient so fast algorithms are needed to find T. An *algorithm*, then, is a procedure which results in a ordered collection of test sets in $\mathcal{X}$ called a *test sequence*:

$$\mathcal{A} \;=\; \{X_1, X_2, ..., X_N\} \;\subseteq \mathcal{X} \quad (N \leq \infty)$$

An algorithm is called *sequential* if the observations

$$f(X_1,T),\; f(X_2,T), ..., f(X_t,T)$$

are used to determine X_{t+1}. Sometimes it is convenient to use the notation

$$Y_t = f(X_t,T)$$

for the t-th observation. If the test sets $X_1,\; X_2, ..., X_N$ are chosen in a way functionally independent of the observations then the algorithm is called non-sequential. Note that in some cases N may be random. An algorithm giving rise to a test sequence $\mathcal{A} = \{X_1, X_2, ..., X_N\}$ is said to be *strongly separating* if the search problem

$$\mathcal{J}^* \;=\; (\mathbf{X}, \mathcal{T}, \mathcal{A}, f)$$

is *always* solveable. "Always" here, acknowledges the fact that $\mathcal{A}$ may depend on T.

Strongly separating means that there is no different $T' \neq T$ leading to the same sequence of observations $\{Y_t\}$. Thus the algorithm distinguishes (separates) the $T \in \mathcal{T}$. Typically the algorithm will expose the true T at the last observation: Y_N .

3. Consistency, predictability, admissibility.

A fundamental idea which pervades much of the methodology for deterministic algorithms is that which is defined here as *consistency*. This is not exactly identified with the concept of a consistency in statistics although there is a strong philosophical connection. Indeed the present notion of consistency is a logical non-probabilistic version of the statistical concept. The idea, here, is that if T is the true target then the values

$$Y_1 = f(X_1, T), ..., \ Y_t = f(X_t, T)$$

logically restrict the class of possible T_j which may remain as candidates to be the target T.

Definition 3. For an algorithm $\mathcal{A}$ and a target T, a set $T_j \in \mathcal{T}$ is said to be *t-consistent* with respect to T and $\mathcal{A}$ if

$$f(X_i, T) \ = \ f(X_i, T_j) \quad (i = 1, 2, ..., t)$$

The set of t-consistent target sets is denoted $\mathcal{T}_t^*$ but it should not be forgotten that it depends on T and $\mathcal{A}$. Note that $T \in \mathcal{T}_t^*$ for all t.

A basic procedure for many algorithms is to eliminate after t observations all $T_j \notin \mathcal{T}_t^*$.

Example 1. Group testing.

Define a search function f as binary if

$$f(X, T) \ = \ \begin{cases} = \ 1 \ \text{ if } X \cap T \ \neq \emptyset \\ = \ 0 \quad \text{otherwise} \end{cases}$$

Then $\mathcal{J}_t^*$ consists of all T_j for which

$$(a) \quad f(X_i, T_j) \ = \ 1 \qquad \text{for all } i = 1, 2, ...t \text{ such that } X_i \cap T \neq \emptyset$$

$$(b) \quad f(X_i, T_j) \ = \ 0 \qquad \text{for all } i = 1, 2, ...t \text{ such that } X_i \cap T = \emptyset$$

Thus considering (a) as defining live sets X_i and (b) dead sets, any T_j which does not intersect a live set or which does intersect a dead set is eliminated.

Elimination above is taken in an "estimation" sense, information is gained by eliminating sets which are definitely no longer candidates to be T. A similar notion

applies to test sets X_i. That is some X_i can be eliminated as not providing any information about T.

Definition 4. A test set X_{t+1} is said to be *t-predictible* for T in $\mathcal{J}$ if the value

$$Y_{t+1} = f(X_{t+1}, T)$$

can be determined without actual observation. That is there is a *known function* H such that

$$Y_{t+1} \;=\; f(X_{t+1}, T) \;=\; H(Y_1, ..., Y_t).$$

By a known function we mean computable from knowledge of $\mathcal{J}$ and $Y_1, ...,\ Y_t$ but without any additional knowledge about T.

The following is a duality theorem connecting t-consistency and t-predictibility. The proof should be taken as applying when $\mathbf{X}$ is finite.

Theorem 1. For a target set T and an algorithm $\mathcal{A}$ the set X_{t+1} is t-predictible if and only if

$$\mathcal{T}_t^* \;=\; \mathcal{T}_{t+1}^*$$

that is, the t-consistent and $(t+1)$-consistent sets are identical.

Proof. Assume X_{t+1} is t-predictible. By t-consistency of $\mathcal{T}_t^*$,

$$f(X_i, T_j) \;=\; f(X_i, T) \quad (i = 1, 2, ..., t)$$

for all $T_j \in \mathcal{T}_t^*$. But then for $T_j \in \mathcal{T}_t^*$

$$\begin{aligned} f(X_{t+1}, T) &= H(f(X_1, T), ..., f(X_t, T)) \\ &= H(f(X_1, T), ..., f(X_t, T_j)) \\ &= f(X_{t+1}, T_j). \end{aligned}$$

Hence $\mathcal{T}_t^* \subseteq \mathcal{T}_{t+1}^*$. But trivially $\mathcal{T}_{t+1}^* \subseteq \mathcal{T}_t^*$ thus $\mathcal{T}_{t+1}^* = \mathcal{T}_t^*$.

Assume that $\mathcal{T}_{t+1}^* = \mathcal{T}_t^*$. Then no more sets are eliminated after observing $Y_{t+1} = f(X_{t+1}, T)$. That is for all $T_j \in \mathcal{T}_t^*$, $f(X_{t+1}, T)$ takes the same value. Since $\mathcal{T}_t^*$ is a *known* collection, that is can be evaluated by checking against $Y_1, ..., Y_t$, selection of any member of $T_j \in \mathcal{T}_t^*$ will yield

$$f(X_{t+1}, T) \;=\; f(X_{t+1}, T_j).$$

The formal definition of H is: $H(Y_1, ..., Y_t) = f(X_{t+1}, T_j)$ for any T_j in $\mathcal{T}_t^*$ for which

$$f(X_{t+1}, T_j) \;=\; f(X_{t+1}, T) \quad (i = 1, ..., t). \qquad \square$$

A more general notion than predictability is that of *admissibility*. To understand this we need to define an idea of "gives more information than" in terms of the structure of the set $\mathcal{T}_t^*$.

Definition 5. A partion $\mathcal{Q}$ of a collection $\mathcal{T}$ is a subdivision into disjoint subcollections: $\mathcal{T} = \mathcal{T}^{(1)} \cup .... \cup \mathcal{T}^{(k)}$. A partition $\mathcal{Q}$ of $\mathcal{T}$ is finer than a partition $\mathcal{Q}'$ if for any $\mathcal{T}^{(i)}$ in $\mathcal{Q}$ there is a $\mathcal{T}'^{(j)}$ in $\mathcal{Q}'$ such that $\mathcal{T}^{(i)} \subseteq \mathcal{T}'^{(j)}$ and for some pair i, j there is strict inclusion.
This idea of a finer partition of is fundamental to the idea that an observation using one particular X is better than with another. An observation induces a partition on $\mathcal{T}$ into blocks within which all T_j have the same observation value $f(X, T)$ and across which have different values.

Definition 6. A test set X'_{t+1} is t-inadmissible (relative to observations $Y_i = f(X_i, T),\ i = 1, ..., t$) if there is a different test set X_{t+1} which induces a finer partition of $\mathcal{T}_t^*$. If the partition of $\mathcal{T}_t^*$ induced by X_{t+1} is either finer than or equal to the partition induced by X'_{t+1} we use the term weakly inadmissible.

Theorem 2.
(i) If there is a t-nonpredictable test set X_{t+1} then any t-predictable set is t-inadmissible.
(ii) Assume that X'_{t+1} is t-inadmissible and an admissible set X_{t+1}, which dominates X'_{t+1} in the sense of Definition 6, is used at time $t+1$, then if there is a $(t+1)$-nonpredictable set X_{t+2} then X'_{t+1} is $(t+1)$-inadmissible.
(iii) If X'_{t+1} is t-inadmissible then it is $(t+1)$-weakly inadmissible (whatever X_{t+1} is used).
The interpretation of this theorem is that, provided improvements continue to be made, once a test set is inadmissible it remains so.

Proof. The proof of (i) is immediate since X_{t+1} is t-nonpredictable if and only if $\mathcal{T}_t^*$ is not partitioned at all. Part (ii) follows since any $(t+1)$-predictable set produces a finer partition than $\mathcal{T}_t^*$ itself, by part (i). But using X'_{t+1} at time $t+1$ cannot produce a finer partition than is generated by $\mathcal{T}_{t+1}^*$ itself since the latter already generates a finer partition. Part (iii) follows a similar argument.

There is a strong relationship here with the statistical notion of maximum likelihood or Bayes rules. In statistics one may consider heuristically what parameter values give the highest (posterior) probability of achieving the given data. If this probability for a given parameter value is actually zero then such a parameter value would under no circumstances be allowable as an estimator. We shall return to this idea in Section 9.

4. Matrix representation.

Matrix representations of search are useful and provide strong connections with experimental design and coding theory. Define the search matrix of a (finite) search

problem $\mathcal{J}$ by

$$S = \begin{bmatrix} f(X_1,T_1) & . & . & . & f(X_1,T_M) \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ f(X_k,T_1) & . & . & . & f(X_k,T_M) \end{bmatrix}$$

the rows of S are indexed by the test field $\mathcal{X}$ and the columns by the target field $\mathcal{T}$. The following is evident.

Proposition 1. A search problem $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ is solvable if and only if all the columns of S are different.

The matrix S, of course, does not need to be calculated explicitly and is usually very large. An algorithm uses only some rows of S, picked sequentially or non-sequentially. The elimination in passing from $\mathcal{T}_t^*$ to $\mathcal{T}_{t+1}^*$ in Section 2 is equivalent to deleting all columns of S incosistent with the existing data. Notice a slight ambiguity of notation between the order in $\mathcal{X}$ and the algorithmic order $X_1, ..., X_t, ...$ of the rows.

Let $\mathcal{X}_t^*$ be the set of t-nonpredictable sets and $\mathcal{T}_t^*$ be the set of t-consistent sets, then let

$$\mathcal{J}_t^* = \left\{ f(X,T) \right\}_{X \in \mathcal{X}_t^*,\, t \in \mathcal{T}_t^*}.$$

The procedure which eliminates any X not in $\mathcal{X}_{t+1}^*$ and any T not in $\mathcal{T}_{t+1}^*$ is interpritated as (a) delete any row of S_t^* with elements all equal and (b) delete any column not consistent with the T-column.

Similarly, for an algorithm let

$$S_N = \left\{ f(X_i,T) \right\}_{i=\,1,...,N;\; T \in \mathcal{T}}$$

be the submatrix for the actual realised test sets $X_1, ..., X_N$. Then the following can be stated.

Proposition 2. Let $\mathcal{J}$ be a solveable search problem. Then a search problem is strongly separating if one and only one column of the submatrix S_N corresponds to the data vector (transposed) $(f(X_1,T), ..., f(X_N,T))^T$.

For a non-sequential algorithms the condition of Proposition 2 should hold regardlees of the test results (i.e. regardless of the true set T) thus a stronger condition is required.

Proposition 3. A non-sequential algorithm is strongly separating if all the columns of S_N are different.

If the consistency approach is taken to eliminate consistent columns as the algorithm procedes then it will solve $\mathcal{J}$ if and only if one column remains, that corresponding to the true target T.

5. Tree Representation.

A convenient way to represent a sequential search algorithm is by a decision tree. In this context a tree consists of *inner nodes, branches* and *outer nodes*, or *leaves*. Inner modes may be thought of as a corresponding to test sets $X_t \in \mathcal{X}$ specified by the algorithm, branches as corresponding to the observed outcomes of these tests, and leaves corresponding to a partition of the target set $\mathcal{T}$.

The search is defined by a direct path through the tree, performing the tests represented by each of the inner nodes encountered and following the branch corresponding to the outcome of the test until a branch leads to a leaf, at which point the search is finished. The leaf represents the subset of the target set $\mathcal{T}$ which is consistent with all of the test results observed along the path through the tree.

Example 2. Bad Penny.

A number of coins are of the same weight except one which is heavier. With the aid of a balance scale the odd coin is to be found with a minimum number of weighings. Figure 1 shows the tree representing the optimal solution for nine coins, where the coins are arbitrarily labelled $A, B, ..., I$.

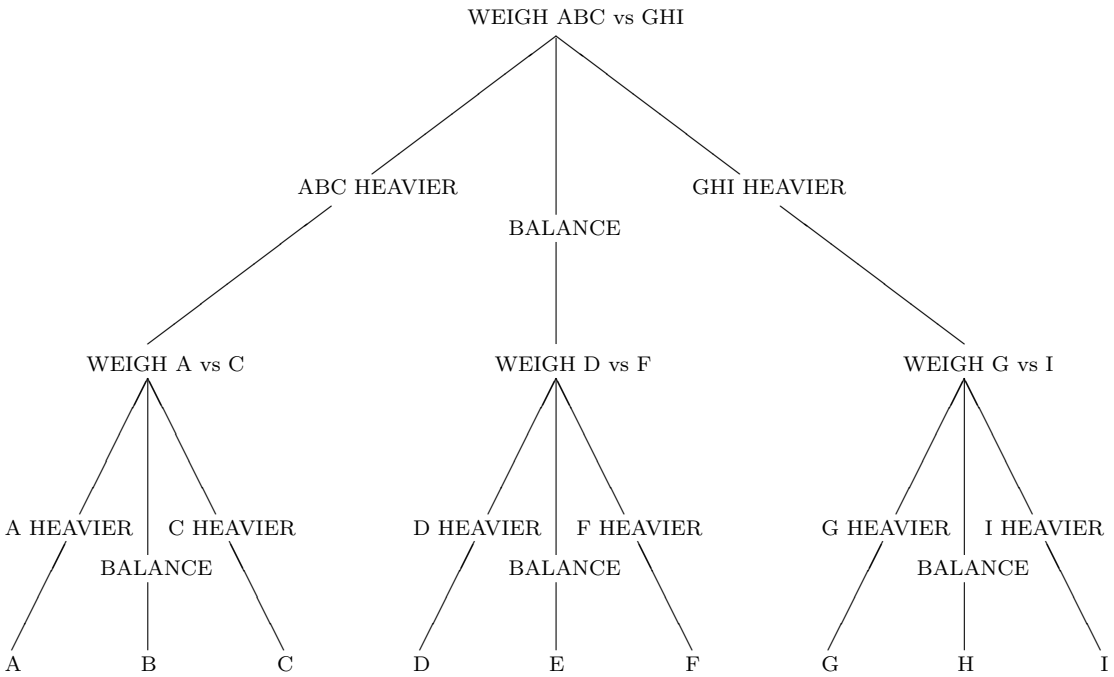


Fig. 1. Optimal tree for bad penny problem.

An inner node corresponding to a test set X_t is said to be the *parent* of all nodes connected to it by the branches corresponding to the possible test outcomes $f(X_t, T)$. A node is its parents' *offspring*. If there is a path leading from a node corresponding to test set X_t to one corresponding to X_{t+k} $(k = 1, 2, ...)$, then the latter is the former's *descendent*. A node is its descendent's *ancestor*. A tree is said to be *rooted* if the following three conditions hold.

(1) There is a single node, called the *root*, which is the ancestor of all other nodes.
(2) No node is its own ancestor (i.e. there are no "loops" in the tree).
(3) Each node other than the root has exactly one parent.

In search terms, condition (1) corresponds to an algorithm having a unique starting point X_1. Condition (2) specifies that a search should not repeat a test or sequence of tests (there are of course search problems where one may wish to repeat observations, for example in the presence of observation errors, but these cases are not discussed here). Although condition (3) demands unique parentage of nodes, it may be the case that two or more nodes correspond to the same test set.

Several properties of a search may be considered in terms of the tree representation. A search is strongly separating if and only if the leaves of the tree correspond to individual target sites $T_i \in \mathcal{T}$ and the union of the sites is $\mathcal{T}$. At time t during the search the consistent set $\mathcal{T}_t^*$ is the union of the target subsets corresponding to the set of leaves which are descendents of the current node. From this and the definition of predictability, it follows that a test set is predictable if and only if its corresponding node has exactly one offspring.

For a strongly separating design suppose that there is a probability distribution on the target sites and write $p_i = prob(T_i = T;\ T_i \in \mathcal{T})\ \ i = 1, ..., M$, so that each leaf can now be considered as having a probability, or *weight*, of being the stopping point of the search through the tree. The *length* of a leaf is defined as the number of branches along the unique path from the root to that leaf. Hence the expected number of tests in the search is $\sum_{i=1}^{M} p_i l_i$ where l_i is the length of the i-th leaf.

Example 3. Alphabetic tree search.

Suppose the target set (and hence the set of leaves on the tree) have some arbitrary or natural ordering, and relabel the elments of $\mathcal{T}$ in such a way that $T_1 < T_2 < ... < T_M$, where "$a < b$" is not necessarily a numerical comparison but rather has the meaning that a comes before b with respect to the ordering. An

alphabetic search is where the test function can be written in the form

$$f(X,T) \;=\; \begin{cases} -1 & \text{if} \quad X < T \\ 0 & \text{if} \quad X = T \\ 1 & \text{if} \quad X > T \end{cases} \tag{1}$$

An example of such a test function is where one is sorting an alphabetically ordered list of files, picking a file at random or according to some rule, and if the target is not found, choosing another file with respect to the previous results. There are two main versions of such a search which may be thought of as retrieval and insertion. When retrieving a file, assuming that the file exists, the test field and target field coincide, i.e. $\mathcal{X} = \mathcal{T}$. All three outcomes of $f(X,T)$ given in (1) are possible in this case.

When inserting a file however, assuming the file does not have the same name as any other file in the list, the target corresponds to a "space" between two files. Suppose there are $M-1$ files in the list. As before the test field is the set of files, label these $X_1, X_2, ..., X_{M-1}$. Then the target field may be written as $\{\ T_1, T_2, ..., T_M\ \}$ where

$$T \;=\; \begin{cases} T_1 & \text{if} \quad T < X \\ T_j & \text{if} \quad X_{j-1} < T < X_j \\ T_M & \text{if} \quad T > X_{M-1} \end{cases} \tag{1}$$

Hence, in this case, $f(X,T)$ as in (1) can only take the values ± 1. It will be seen in Section 10 that the nested binary group testing problem can be considered as a sequence of alphabetic retrieval searches.

For the retrieval case, suppose that $prob(\,T = T_i\,) \;=\; p_i$, hence $\{p_i\}_{i=1,...,M}$ form a set of weights on the leaves of the search tree. Then the following "bottom up" tree construction gives the optimum algorithm (in the average case) for solving this problem.

Step 1. Start with outer nodes $\boxed{p_1}$, $\boxed{p_2}$, ..., $\boxed{p_M}$. Combine the nodes i and j are chosen according to the following rules:

(i) No end-node occurs between i and j.
(ii) The sum $p_i + p_j$ is minimum among all (i,j) satisfying (i).
(iii) The index i is minimum over all (i,j) satisfying (i), (ii).
(iv) The index j is minimum over all (i,j) satisfying (i), (ii), (iii). This is

repeated until only one node remains, i.e. a rooted tree has been constructed. Each outer node is then assigned its length l_i as its weight. Inner nodes are discarded and a new tree is constructed by combining nodes i and j where i and j satisfy the following rules:

(i) The nodes i, j are adjacent in the sequence.
(ii) The lengths l_i, l_j are maximal.
(iii) The index i is minimal over all (i, j) satisfying (i), (ii).

6. Belief measures and random T .

It will turn out that a convenient way of generating good algorithms is to let the choice of the test set X_{t+1} depend on our state of knowledge or belief. Our belief at time t or after t observations is measured, then, by a *belief function* μ_t on $\mathcal{B}(\mathcal{T})$, that is a collection of sets in $\mathcal{T}$. This represents our current beliefs about the possible candidates for T. The notation $\mu_t(\{T_i\})$ refers to the value of the function at the set $\{T_i\}$ constructed now as a single point in $\mathcal{B}(\mathcal{T})$. Thus for example, $\{T_1\} \cup \{T_1, T_2\}$ is the pair $\{T_1\}$ and $\{T_1, T_2\}$ and is not to be confused with say, $T_1 \cup T_2$ or $\{T_1, T_2\}$. The advantage of placing the belief function on $\mathcal{B}(\mathcal{T})$ rather than, say, on $\mathbf{X}$ itself will become clearer as we procede.

Example 4. Alphabet.
Let $\mathbf{X}$ be the letters of the alphabet *a,b, ...,z*. Let $\mathcal{T}$ be the collection of all target words and T be a target word. Then if μ_t is a measure on $\mathcal{B}(\mathcal{T})$ this can be consideredd as a function on the collection of words. It represents for a particular collection of words the belief that T lies in the collection.

The following axioms will often be assumed, but will be made clear at the time.

(i) (normalization) For all $T \in \mathcal{T}$, $0 \leq \mu_t(\{T\}) \leq 1$.
(ii) (elimination) For any $B \in \mathcal{B}(\mathcal{T})$, $\mu_t(B) = 0 \Rightarrow T \notin B$.
(iii) (achievment) $\mu_t(\{T_j\}) = 1$ and $\mu_t(\{T_j\}) = 0$ for all $T_i \in \mathcal{T}$, $T_i \neq T_j$, $\Rightarrow T = T_j$.
(iv) (additivity) For disjoint B_1, B_2 in $\mathcal{B}(\mathcal{T})$, $\mu(B_1 \cup B_2) = \mu(B_1) + \mu(B_2)$.

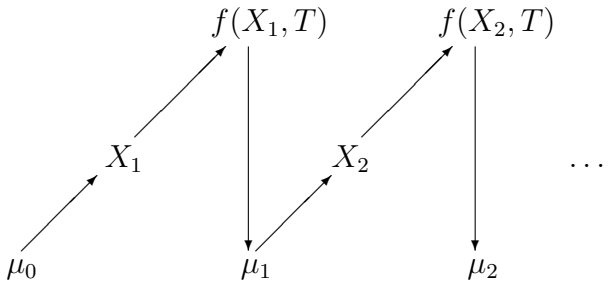
It transpires that the measure μ_t is a convenient way to put certain sequential and non-sequential algorithms into the same framework.

For the non-sequential algorithms defined in the next section one is only concerned with μ_0, a start measure which defines a prior assumption in the location of T. For example if each unit x is in T independently with probability p (the Bernoulli assumption) then this clearly induces a belief measure on T:

$$\mu_0(T) = p^{|T|}(1-p)^{M-|T|}$$

where $|T|$ is the size of T. For non-sequential algorithms all probabilities are then evaluated with respect to μ_0.

For Bayes, entropy based and similar strategies a typical algorithm will look like this:



There are two decision rules to study. The first is to select a new test set based on current beliefs

$$\mu_t \rightarrow X_{t+1}$$

and then to update the beliefs using the new observation $f(X_{t+1}, T)$. A warning is due at this point. It is to be expected that μ_t should summarise all the current information about the "state" of the system. However it is a very familiar fact that this is over simplistic. It is often very difficult to achieve a globally optimum sequential procedure with a simple one-step-ahead algorithm. This complicates many sequential decision procedures in sequential experimentation, dynamic control and AI.

7. Non-sequential algorithms.

It turns out that the matrix representation of Section 3 is particularly useful in assessing non-sequential measure μ_0 is taken to be a proper probability measure.

Definition 7. A search algorithm leading to a *design* $A = \{X_1, \ldots, X_N\}$ is called γ-separating with respect to an initial probability measure μ_0 if

$$\mu_0(|\mathcal{T}_N^*| = 1) \geq (1 - \gamma) \qquad (0 \leq \gamma < 1)$$

where $\mathcal{T}_N^*$ is the N-consistent set for A.

The quantity $\mu_0(|\mathcal{T}_N^*| = 1)$ is be interprited as the probability that the final consistent set of the algorithm contains only one member (the target). Therefore a γ-separating algorithm finds the target with probability not less than $1 - \gamma$. A γ-separating algorithm in general is called weakly separating. A strongly separating algorithm then, as defined in Section 2 is 0-separating in the present terminology and because of the inequality in Definition 7 is γ-separating for any $\gamma > 0$. For strongly separating algorithms $|\mathcal{T}_N^*| = 1$ with probability one. Usually 0-separating disigns are also strongly separating.

It is sometimes the case that for a search algorithm the γ-separating property is defined with a probability measure other than μ_0. This is typically the case when the algorithm itself is random, that is the search introduces a probability in selection

of the test sets X. Great care must be taken in reading the literature in establishing where the probabilities come from and distinguishing between prior/posterior probabilities and randomization in the choice of test set.

There are several reasons for the popularity of random search in the sense of selection of X. Here are two. First, random algorithms may be useful when the construction of optimal or good design is too complex. Roughly a random search may have reasonably "space-filling" features. Second, if bounds for $\mu_0(|\mathcal{T}_N^*| = 1)$ or $prob(|\mathcal{T}_N^*| = 1)$ can be found under a randomization scheme then the existence of at least one *deterministic* algorithm which achieves the bound can be inferred. The following methodology originated in the work of A. Renyi in establishing the existence of γ-separating algorithms with given "sample size" N in many combinatorial search problems.

Let $\mathcal{X}^N = \left\{\{X_1, \ldots, X_N\},\ X_i \in \mathcal{X}\right\}$ be a certain family of N-collections of test sets. The family consists of the test sequence for various search algorithms with sample size N. Let $\mathcal{P}_N$ be a probability measure on $\mathcal{B}(\mathcal{X}^N)$ and let ξ_N be the associate random vector in $\mathcal{X}^N$. The vector ξ_N can be considered as giving a realized algorithm under $\mathcal{P}_N$. The Renyi method is formalised as follows.

Proposition 4 (existence principle). If for some N and P_N on $\mathcal{X}^N$,

$$P_N = prob\{\xi_n \text{gives a } \gamma\text{separating algorithm}\} > 0$$

then at least one (non-random) γ-separating algorithm with sample size N exists.

Notice here that the outside probability is computed with respect to the sampling scheme $\mathcal{P}_N$ but that μ_0 is still used in the γ-separating definition. The proof of Proposition 4 is transparent. Consider a sequence $\{\xi_{N,i}\}$ of independent realizations of random ξ_N and define

$$\beta_{N,i} = \begin{cases} 1 & \text{if } \xi_{N,i} \text{ leads to a } \gamma - \text{separating algorithm} \\ 0 & \text{otherwise .} \end{cases}$$

Then, since $0 \leq \gamma < 1$,

$$Prob\{\beta_{N,i} = 1\} = p > 0$$

so that with probability one at least one member of $\{\xi_{N,i}\}$ corresponds to a γ-separating algorithm.

The art, then is to choose $\mathcal{P}_N$ to make the probability P_n in Principle 4 large enough to establish useful lower bounds of the form

$$P_N > l_N.$$

If $\mathcal{P}_N$ can be found such that $l_n \geq 0$ then a γ-separating algorithm exists. Alternatively one can have

$$P_N \geq l_N > 0,$$

achieving the same result. Typical choices for P_N are uniform on $\mathcal{B}(\mathcal{X}^N)$ and sampling without replacement from $\mathcal{X}^N$.

The following theorem generalizes some results of A.Renyi.

Theorem 3. Let $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ be a solveable search problem. Let $|T| = M$ and $|\mathcal{X}| = R$. Let k_{ij} $(i, j = 1, 2, \ldots, M)$ be the number of $X \in \mathcal{X}$ such that

$$f(X, T_i) \;=\; f(X, T_j).$$

Then there exists a non-sequential strongly separating search algorithm with sample size

$$N \;\leq\; \min\left\{ n : \sum_{i=1}^{M} \sum_{j=1}^{i-1} \left(\frac{k_{ij}}{R}\right)^n \;\leq 1, \quad n = 1, 2, \ldots, \right\},$$

Proof. The key to the proof is to select N large enough so that there is positive probability of the columns of the corresponding matrix S in the notation of Section 4 being distinct.

Let N be fixed and let $\mathcal{P}_N$ be the uniform distribution on $\mathcal{B}(\mathcal{X}^N)$, i.e. the product of uniform distribution on $\mathcal{X}$. Thus $X_1, \ldots, X_N$ are independent uniformly distributed on $\mathcal{X}$. This can also be thought of classical simple random sampling with replacement. It implies that the rows of $A = \{f(X_i, T_j)\}$ are indepently and identically distributed. Denoting the columns of thematrix by $a_1, \ldots, a_m$. Then

$$\begin{aligned} prob\{a_i = a_j\} \;&=\; prob\{f(X_l, T_j) \text{ for } l = 1, \ldots, N\} \\ &= \prod_{l=1}^{N} prob\{f(X_l, T_i) = f(X_l, T_j)\} \;=\; \prod_{l=1}^{N} (k_{ij}/R) \;=\; (k_{ij}/R)^N. \end{aligned}$$

Now the probability that the matrix S gives does not give a strongly separating algorithm is

$$prob\{a_i \;=\; a_j \text{ for some pair } (i, j)\; i < j\} \;\leq\; \sum_{i<j} \sum (k_{ij}/R)^N.$$

Thus the probability that the algorithm *is* strongly separating is not smaller than

$$1 \;-\; \sum_{i<j} \sum (k_{ij}/R)^N,$$

yielding the result. □

Corollary 3. Let the conditions of Theorem 1 hold and let

$$k \;=\; \max_{i \neq j} k_{ij}.$$

Then
(a) There exists a non-sequential strongly separating algorithm with sample size

$$N \;\leq\; \left[\frac{\log\left(\frac{R(R-1)}{2}\right)}{\log\left(\frac{R}{k}\right)}\right] \;+\; 1$$

where [.] means the integer part.

The following technique can be used to establish *lower* bounds for the lenght of strongly separating algorithms. First, it is clear that the number of possible answers in N experiments should not be smaller than the number of targets. This immediately gives

$$L^N \;\geq\; R,$$

where L is the maximal number of values which the test function $f(.,.)$ can have. In some cases the corresponding lower bound for N,

$$N \;\geq\; \frac{\log R}{\log L},$$

can be improved using the structure of the search problem. Suppose, for instance, that the collection of test results $\{\, f(X,T)\,\}_{t\in\mathcal{T}}$ coincide for all $X \in \mathcal{X}$. Then for any s $(0 \leq s \leq L)$ we can ignore $k(s)$ elements from $\mathcal{T}$ corresponding to some s test results, supposing that the corresponding elements are separated. Then we obtain

$$(\,L \;-\; s\,)^N \;\geq\; R \;-\; N\,k(s).$$

In particular, in the game of mastermind described in Section 11 for $m \geq 3$ (in the notation of that section) we can choose $s = 1$, $k(s) = 1$ and always improve the lower bound.

The second type of lower bound is based naturally on entropy (see Section 9). Letting h denote the maximal entropy contained in a single observation and H the total entropy of the system we use the subadditivity of entropy to obtain: $H \leq Nh$ or

$$N \;\geq\; \frac{H}{h}.$$

8. Observer logics.

Search routines are built into the whole AI machinery. Languages such as PROLOG and systems too numerous to refer to contain tree or graph search methods of various kinds. Various methodologies have cores which are search-like and one can

mention: computational learning theory, dynamic decision processes, mechanical theorem proving and so on.

One way to tie together the set-based formulation of search in this paper is directly with the propositional calculus of logic in the following way. Let $\mathbf{X}$ be a collection of primitive propositions. We want as our target the collection of true propositions, but this is a little inexact. Let T be the collection of true propositions by which we mean that the proposition

$$\tilde{T} \;=\; \wedge_{x\in T} x \quad \wedge_{x\notin T} \neg\, x$$

is true. Thus for any $T_j \;\neq T$ the statement

$$\tilde{T_j} \;=\; \wedge_{x\in T_j} x \quad \wedge_{x\notin T_j} \neg\, x$$

is false.

Considering the binary case of Example 1 a set X is used to test for T in the following way. Let

$$\tilde{X} \;=\; \vee_{x\in X}\, x$$

Then let $f(X,T) \;=\; 1$ if

$$\tilde{X} \wedge \tilde{T} \;=\; \left(\vee_{x\in X}\, x\right) \;\wedge\; \left(\; \wedge_{x\in T}\, x \wedge \;\wedge_{x\notin T} \neg\, x\right)$$

is the true and $f(X,T) \;=\; 0$ otherwise. We shall abuse the notation here by identifying T with $\tilde{T}$ and X with $\tilde{X}$.

Now the logic of search is dynamic in the following sense. The set of true propositions is unknown. Observations are the declaration of the truth or falsehood of new specimen propositions such as $\tilde{X} \wedge \tilde{T}$. It is worth working through the consistency / non-predictability steps of Theorem 1 in terms of truth tables.

If $\mathcal{T}_t^*$ is the consistent set after t steps then $\mathcal{T}_t^*$ in terms of the logic contains all sets T_j which are still "candidates" for the set of true propositions, that is cannot be eliminated purely on the basis of statements up to t steps. Within $\mathcal{T}_t^*$ let $T_\circ$ be the true target (set of true targets). In the following table the observation is the second column, the third and fourth columns being implied.

T_0	$X \wedge T_0$	X	$X \vee T_0$
T	T	T	T
T	F	F	T

Table 1.

Now consider a second table in which the results of the first table are taken as inputs.

X	$X \wedge T_j$	$X \vee T_j$	T_j
T	F	T	F
F	F	F	F

Table 2.

Since X true implies $X \vee T_j$ true and X false implies $X \wedge T_j$ false, two of the cells in Table 2 are obtained directly from the input values of X. However, if the ringed cells yield "false" then T_j can be eliminated as false, i.e. $\neq T$. Following Theorem 1 it is seen that predictability means that for no $T_j \in \mathcal{T}_t^*$ do the ringed cells yield false that is each such T_j acts to yield a "true" making it indistinguishible from $T_\circ$ itself.

Now turn to the idea of a belief function. The most obvious is the indicator function

$$\mu_t(\{T\}) \;=\; \begin{cases} 1 & \text{if } t \in \mathcal{T}_t^* \\ 0 & \text{otherwise.} \end{cases}$$

This can be extended to all of $\mathcal{B}(\mathcal{T})$ by additivity.

In the context of the logical approach in which $\mathcal{X}$ and $\mathcal{T}$ are possible statements in the same propositional calculus it is natural to take a 3-valued logic

$$\begin{aligned}
\tilde{\mu}_t(A) &= 1 \qquad \text{if it is known that } A \text{ is true} \\
&= 0 \qquad \text{if it is known that } A \text{ is false} \\
&= c \qquad (0 < c < 1) \text{ when the truth or falsehood of } A \\
&\qquad\quad \text{is not decideable from the data.}
\end{aligned}$$

An "observer logic" is then a method of choosing X_{t+1} and updating $\tilde{\mu}_t(A)$ to $\tilde{\mu}_{t+1}(A)$ for every A consistent with the rules of propositional calculus. An observation has the status that

$$\tilde{\mu}_t(X_i) = 1 \text{ or } 0 \qquad (1 \leq i \leq t),$$

and X_{t+1} is predictable if

$$\tilde{\mu}_t(X_{t+1}) = 1 \text{ or } 0.$$

Similarly $\mathcal{T}_t^*$ is the set of T_j for which

$$\tilde{\mu}_t(T_j) = c$$

and $\mathcal{X}_t^*$ (the set of unpredictable X_j) includes those for which

$$\tilde{\mu}_t(X_t^*) = c.$$

At the last step $\tilde{\mu}_t(T) = 1$ for exactly one $T \in \mathcal{T}_t^*$ and $= 0$ otherwise. This approach can be used with any interrogating system for which interrogation leads to the truth or falsehood of a proposition. An alternative to the binary test would be an inclusion test

$$f(X,T) = \begin{cases} 1 & \text{if } T \subseteq X \\ 0 & \text{otherwise.} \end{cases}$$

Taking the interpretation of T and X as before

$$\begin{aligned}
\{f(X,T) = 1\} &\iff \{\neg\,(T \wedge \neg\, X)\} \\
&\iff \{T \to X\}
\end{aligned}$$

similarly

$$f(X,T) = 0 \iff (T \wedge \neg\, X).$$

Finally a deeper interpretation of a solveable search algorithm is as a method of *proving* that T contains exactly all the true statements, mechanically. A request

to use a test set X_t is merely the incorporation of a line in the proof. Solving a search problem is then equivalent to writing down a proof of every true statement. Conversely if a search problem is not solveable then there is some choice of T for which the truth of T cannot be decided given the test sets $\mathcal{X}$. Going further one can interpret the declaration of the truth value of a test set X as an *axiom*. This becomes more useful when the statements being tested are not just the truth of propositions but the truth of statement incorporating any connectors including $\neg$, $\rightarrow$ and brackets. Although there is no difficulty in the core of finite alphabets and first order logics one can see that deep problems of consistency and predictability may arise. If an observation is a new axiom, what guarantees that the new system is consistent or complete?

9. Bayes and Entropy Methods.

A simple way to formulate the Bayes version of the search problem

$$\mathcal{J} \;=\; (\mathrm{X}, \mathcal{T}, \mathcal{X}, f)$$

is to assign prior, and then posterior, probabilities to the inclusion of a particular x (in $\mathbf{X}$) in the target T. This can be done easily with indicator functions (at time t)

$$\begin{aligned} I_t(x) \;&=\; 1 \qquad \text{if } x \in T \\ &=\; 0 \qquad \text{otherwise.} \end{aligned}$$

The statement $T = T_j$ is then equivalent to

$$\begin{aligned} I_t(x) \;&=\; 1 \qquad \text{if } x \in T_j \\ &=\; 0 \qquad \text{otherwise.} \end{aligned}$$

The collection of indicator functions $I_t(x)$ induces a measure on $\mathcal{B}(\mathcal{T})$ in a straightforward fashion. If T_i is considered as a point in $\mathcal{B}(\mathcal{T})$ then define

$$\begin{aligned} \mu_t(\{T_j\}) \;&=\; prob_t(T_j = T) \\ &=\; prob_t(I_t(x) = 1,\; x \in T_j;\; I_t(x) = 0,\; x \notin T_j) \\ &=\; prob_t(\textstyle\prod_{x \in T_j} I_t(x) = 1) \end{aligned}$$

The extension to the whole of $\mathcal{B}(\mathcal{T})$ is by additivity

$$\mu(B) \;=\; \sum_{\{T_j\} \subseteq B} \mu_t(\{T_j\}).$$

At this point there is a serious question of interpretation. One view might consider T as fixed and the indicator functions as being updated in the Bayesian fashion. The other view would be to consider T as fixed and the indicator functions as being updated in the Bayesian fashion. The other view would be to consider T itself as random with distribution given by μ_t at time t and the indicator function as merely induced by μ_t. We are inclined in this section to take the second view as it gives an easier conceptual development of a classical Bayes formulation. Note that when $\mathcal{T}$ is continuous a full random set development is necessary. in that case the random indicator process $I_t(x)$ induces a random set theory on $\mathbf{X}$.

Let μ_0 be the prior distribution arising from the initial joint distribution of the $I_t(x)$. The Bayesian updating step is to condition on the new observation $f(X_{t+1}, T)$. Note that in what follows we write $T = T_i$ to mean the statement that the *random variable* T takes the value T_i.

If all the probabilities are (already) conditional on the current data

$$y_i \;=\; f(X_i, T) \qquad i = 1, \ldots, t$$

then

$$\mu_{t+1}(T_j) \;=\; (\, prob(T = T_j \,|\, f(X_{t+1}, T) \;=\; y_{t+1})$$

$$= \; \frac{prob(\{T = T_j\} \;\cap\; \{f(X_{t+1}, T) \;=\; y_{t+1}\})}{prob\{f(X_{t+1}, T) \;=\; y_{t+1})\}}$$

$$= \; \frac{prob(\{T = T_j\} \;\cap\; \{f(X_{t+1}, T) \;=\; y_{t+1}\})}{\sum_i \; prob(f(X_{t+1}, T_i) \;=\; y_{t+1})\mu_t(T_i)}$$

Following a classical decision-theoretic formulation we introduce a loss function $L(T, \hat{T})$ which determines a loss on taking the action $\hat{T}$. This action is interpreted as a declaration that $T = \hat{T}$. At time t the Bayes risk conditional on $Y_1, \ldots, Y_t$ for a new observation Y_{t+1} is

$$R_{t+1}(\hat{T}) \;=\; E_{Y_{t+1}} \; E(L(T, \hat{T})|Y_{t+1})$$

where $Y_{t+1} \;=\; f(X_{t+1}, T)$. The inside expectation is conditional on $Y_1, \ldots, Y_{t+1}$. The Bayes rule is to select $T = \hat{T}_B$ to achieve

$$\min_{\hat{T}} E(L\,(T, \hat{T})|\, Y_{t+1})$$

namely the choice of $\hat{T}$ is to minimise the posterior risk given $Y_1, \ldots, Y_{t+1}$.

The Bayes selection of X_{t+1}, namely the Bayes sampling procedure is to select X_{t+1} which achieves

$$\min_{X_{t+1}} R_{t+1}(\hat{T}_B)$$

That is to minimise with respect to X_{t+1} the Bayes risk, or equivalently the prior expectation of the posterior risk.

Various loss functions may be used. For example

$$\text{(i)} \quad L(T,\hat{T}) \;=\; 0 \quad \text{if } T=\hat{T}$$
$$\;=\; 1 \quad \text{otherwise.}$$

(zero-one loss)

$$\text{(ii)} \quad L(T,\hat{T}) \;=\; |T \triangle \hat{T}|$$

where $T\triangle\hat{T}$ is the symmetric-difference $T\triangle\hat{T} \;=\; (T\cap\overline{\hat{T}}) \cup (\overline{T}\cap\hat{T})$.

Example 5. *Bernoulli.* Consider the case where each indicator $I_0(x)$ is independent Bernoulli with $prob(I_0(x)=1)=p$. Let $f(X,T)$ be as in Example 1, and take the loss function (i) above. Consider the first stage update from μ_0 to μ_1. Suppose the first data value is

$$y_1 \;=\; f(X_1,T) \;=\; 1.$$

Note that

$$\mu_0(T_j) \;=\; prob(T=T_j) \;=\; p^{|T_j|}(1-p)^{|\mathbf{X}|-|T_j|},$$

and under μ_0

$$prob(f(X_1,T)=1) \;=\; prob(T\cap X_1 \;\neq\; \emptyset)$$

$$\;=\; 1 \;-\; (1-p)^{|X_1|}.$$

Thus

$$\mu_1(T_j) \;=\; \frac{(1-(1-p)^{|X_1|})\,p^{|T_j|}\,(1-p)^{|\mathbf{X}|-|T_j|}}{\sum_i (1-(1-p)^{|X_1|})\,p^{|T_i|}\,(1-p)^{|\mathbf{X}|-|T_i|}}$$

and

$$\mu_1(T_j) \;=\; 0 \quad \text{for } X\cap T_j=\emptyset.$$

For succesive values of $f(X_t,T)$ the elimination of T_j is as for the non-Bayesian case and the remaining T_j have their probabilities *renormalised*.

In this example, with the (0-1) loss function, (i) the Bayes action $\hat{T}$ is to select the T_j with the highest posterior probability and the sampling rule X_{t+1} is to make the expected value of this as high as possible. This could be described as a "*greedy*" algorithm.

An approach is possible which does not explicitly use a loss and risk function but rather a measure of dispersion directly on μ_t. The best known of these is entropy defined as

$$Ent\,(\mu_t) \;=\; -\sum_{T_i} \mu_t(T_i)\,\log\,\mu_t(T_i).$$

This can also be considered as one definition of the entropy of the family of indicator functions $\{I_t(x)\}$ at time t. A useful equivalence exists between this entropy and

that of $\{f(X_t, T)\}$ the collection of observations as X_t ranges over all $\mathcal{X}$. Clearly $\{f(X_t, T)\}$ is functional dependent on T and hence on $\{I_t(x)\}$. If every T in $\mathcal{T}$ is achievable (that is if the original search problem $\mathcal{J}$ is solveable) then every $I_t(x)$ is functional related to $\{f(X_t, T)\}$. For example if $f(X_t, T)$ is binary then a measure m_t is induced on $\mathcal{X}$ and

$$Ent\,(\mu_t) \;=\; Ent\,(m_t).$$

There is a convenient formula for updating the entropy on observing $y_{t+1} = f(X_{t+1}, T)$.

$$Ent\,(m_t) \;=\; Ent\,(y\,(X_{t+1}, T)) \;+\; E_{y_{t+1}}(Ent\,(m_t|y_{t+1})).$$

So that

$$Ent\,(\mu_t) \;=\; Ent\,(y_{t+1}) \;+\; E_{y_{t+1}}(Ent\,(\mu_t|y_{t+1})).$$

The quantity $Ent\,(\mu_t|y_{t+1})$ is just $Ent\,(\mu_{t+1})$ the conditional entropy given all observations up to and including y_{t+1}. The full second term on the right hand side is the prior expectation of this quantity, the direct entropy analog of the preposterior Bayes risk. At time t the entropy $Ent\,(m_t)$ is fixed. Thus to minimise the preposterior entropy (second term) it is necessary and sufficient to maximise the first term over the choice of X_{t+1}. Here of course the entropy is conditional on $y_1, \ldots, y_t$. In the Binary case this step is achieved if, conditional on $y_1, \ldots, y_t$,

$$prob(\; f(X_{t+1}, T) \;=\; 1\;) \;=\; \frac{1}{2}$$

$$prob(\; f(X_{t+1}, T) = 0) \;=\; \frac{1}{2}.$$

This represents a generalised bifurcation rule and also an upper bound on the amount of entropy reduction possible at every t.

Finally, in this section the minimax sequential rule should now be clear. It is similar to the entropy rule except the criterion if different name select X_{t+1} to minimise

$$\max_{B \in \mathcal{Q}} |B|.$$

where $\mathcal{Q}$ is the partion induced in $\mathcal{T}_t^*$ by X_{t+1}. Thus we attempt to make the partition finer not in the sense of entropy but with respect to the size of the largest block among in the partion of the consistent sets.

10. Screening and Group Testing.

Consider the case where some response function depends on a large number of controllable factors, but where it is known (or suspected) that only a few of these factors actually have any significant effect on the response. By observing

the response from tests involving groups of factors (group tests), we aim to find the important ones using as few tests as possible. Such an experiment is called *screening*.

To formulate the problem in the search notation, denote the factors $x_1, x_2, \ldots, x_n$, thus the set of search units is $\mathbf{X} = \{x_i\}$. A test group consists of some (or any) combination of factors, so that $\mathcal{X} \subseteq \mathcal{B}(\mathbf{X})$. Let the target T be the set of important factors so that $\mathcal{T} \subseteq \mathcal{B}(\mathbf{X})$. The search function is the response, written $f(X, T)$.

The group testing problem consists of two main cases, those with a binary response and those with an additive response. These two cases will now be discussed along with several possible algorithms.

Binary group testing.

This was discussed earlier in example 1. To recap, the population consists of dichtomous units, satisfactory and defective say, and the response is binary and of the form

$$f(X,T) \;=\; \begin{cases} 1 & \text{if } X \cap T \neq \emptyset \\ \\ 0 & \text{otherwise} \end{cases}$$

where the target T is the set of defective units. A group which is known to contain at least one defective is called a defective group. Example 3 considered the Bayesian updating of this problem when there is a prior probability distribution over the target set $\mathcal{T}$. In particular, the binomial case was considered where units are independently defective or satisfactory with probabilities p and $q = 1 - p$ respectively.

We now consider two algorithms which have particular interest to search procedures in general.

Consider first the maximum entropy algorithm. Example 3 stated that the amount of entropy in a group test is maximised by choosing the test group X such that the probability of either outcome of the test function $f(X, T)$ is as close as possible to $\frac{1}{2}$, i.e. minimise

$$|\frac{1}{2} - \sum_j f(X, T|\, T = T_j)\, \mu(T_j)| \tag{2}$$

where in this case the value of the belief function at a target site is the probability of the target being at that site, i.e. $\mu(T_j) = prob(T = T_j)$. However, the amount of computation required to compute (2) for each choice of X_{t+1} is usually prohibitively large. Although at the begining of the search the independence and symmetry of the factors of the probability distribution over $\mathcal{T}$ make it possible to minimise (2) explicity, as the search progresses and the distribution is updated with respect to the test results, μ generally becomes increasingly complex. One solution is to search for a test set such that $prob(f(X, T) = 1)$ is "sufficiently" close to $\frac{1}{2}$. This along

with several other properties of the problem, reduces the amount of computation required.

An alternative way of approximately minimising (2) which achieves great reductions in computation is non-mixing. This involves imposing constraints, dependent on previous results, on the choice of the next test set. Non-mixing uses the following result:

Lemma 1. (Sobel and Groll) Suppose a binomial group $\mathcal{A} + \mathcal{B}$ is tested giving a positive result, and a further test on $\mathcal{A}$ only also gives a positive result. Thus $\mathcal{A}$ forms a defective group. Then the updated distribution of $\mathcal{B}$ conditional on these two results is the same as the original binomial distribution.

The proof of this lemma is easily derived using Bayes rule.

Suppose we impose the restraint (known as *non-mixing*) on each choice of test group that it must be a subset of a defective group if one exists at that stage. Then by the above lemma, at each stage all unclassified units belong to one of only two sets, a defective set or a binomial set. Then, since within each set units are identically distributed, the problem is reduced only to finding the size of the next test group. Further computational savings are achieved with the use of bounds on the test group size obtained by. However, the test group size may be found immediately to within ± 1 by simply considering the probability of either outcome, as follows: In the binomial case, if $|X|$ denotes the number of units in the group X, then

$$prob(f(X,T) = 0) \;=\; 1 - q^{|X|}.$$

Setting this to $\frac{1}{2}$ gives

$$|X| \;\approx\; \frac{\log \frac{1}{2}}{\log q}.$$

In the defective case, let D be the defective set from which the test group is to be chosen so that $X \subset D$. Then

$$prob(f(X,T) = 0) \;=\; \frac{1 - q^{|X|}}{1 - q^{|D|}}$$

and setting this to $\frac{1}{2}$ gives

$$|X| \;\approx\; \frac{\log(\frac{1}{2}(1 - q^{|N|}))}{\log q}.$$

In either case, the integer just above or just below the given value is the optimum group size. Once the size of the test group has been determined the group may be chosen arbitrarily from within the relevant set (i.e. binomial set or defective set).

Thus it is seen that sequentially restraining where the search "looks next", large computational savings may be achieved.

We now consider an optimal non-mixing procedure. Suppose that the units are arbitrarily ordered $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ and that, without loss of generality, when choosing a test group of size k (where k is determined by the algorithm used), the group is formed from the first k unclassified units (with respect to the ordering). Then it is clear that provided k does not exceed the size of the defective set if one exists,or the size of the binomial set if the defective set is empty (this condition is by definition automatically satisfied by any non-mixing algorithm), then the non-mixing constraint is satisfied. Now it is intuitively clear (by virtue of lemma (1)) that if there are any defectives in the population then a non-mixing algorithm carried out in this manner will reach a point where the defective group consists of a single unit, $x_{(i)}$ say. This unit will be the first defective unit in the population (with respect to the ordering). At this point the units $x_{(1)}, \ldots, x_{(i-1)}$ will have been classified as satisfactory whereas the units $x_{(i+1)}, \ldots, x_{(n)}$ will have the original binomial distribution. Note that this is true regardless of what non-mixing algorithm was used. The unclassified (binomially distributed) units $x_{(i+1)}, \ldots, x_{(n)}$ will then also be broken down until the first defective is found (if one exists) and so on until the whole population has been classified. Thus it can be seen that a non-mixing algorithm can be problematically reduced to a series of searches within decreasing binomial sets for the first defective unit.

Now consider the binomially distributed ordered units $x_{(1)}, \ldots, x_{(n)}$ as the leaves of a search tree and consider the problem of searching for the first defective amongst these leaves. A group test consists of testing the first k units $x_{(1)}, \ldots, x_{(k)}$ where k is the test group size. Suppose now that there is an imaginary point, x^* say, between the leaves $x_{(k)}$ and $x_{(k+1)}$. Then if the group test $f(\{x_{(1)}, \ldots, x_{(k)}\}, T)$ is positive then the first defective lies before x^* (with respect to the ordering), whereas if the test is negative then the first defective lies after x^*. The search for the first defective can therefore be viewed as an alphabetic search, choosing (imaginary) points x_t^* say and observing whether the target (the first defective) lies to the right or left of that point. As was seen in Section 5, an alphabetic search can be optimally solved using a tree construction algorithm. Hence we see that the optimal non-mixing algorithm consists of a sequence of optimal alphabetic trees.

Additive group tessting and hierarchical procedures.

Another version of the screening problem is the case where important factors contribute an additive effect towards the response. This may be considered for example as an amount of "bad" ingredient present in defective units.

The main difference between this model and the binary model is that if a test is performed on a subgroup of a group which has already been tested, then the difference between the two response observations provides equal information about the complement of the subgroup (within the original group). However, the updating of μ_t is complicated since all previous results must be considered when updating at

each stage. This problem can be overcome by considering the class of hierarchical procedures, wherein two factors may be tested together in a group only if each previous test group contained neither or both of the factors. In this case only the outcome of the most previous test containing all members of the current group need be considered when updating μ_t.

It may be that there is some prior knowledge about the additive model, for example aprior distribution on $\mathcal{T}$ (where $T \in \mathcal{T}$ is as usual the set of important factors), or alternatively there may be some prior knowledge about the size of the additive effect (or amount of "badness") of effective factors. Consider again the binomial case with factors independently effective or non-effective with probabilities p and $q = 1 - p$ respectively, and suppose that effective factors have a positive but arbitrary effect. Let $N^*(r)$ be the expected number of tests required to classify r factors using an optimal hierarchical procedure. Then $N^*(r)$ may be expressed recursively as

$$N^*(r) \;=\; -q^r \;+\; \min_{1 \le k \le r} \{N^*(k) + N^*(r-k)\} \tag{3}$$

with boundary condition $N^*(1) = 1$. Moreover, the minimising value of k in (3) is also the optimum test group size if the r factors form a defective group. Furthermore, the recursive equation

$$F(n) \;=\; f(n) \;+\; \min_{1 \le k \le n} \{F(k) \;+\; F(n-k)\} \tag{4}$$

where $F(1) \;=\; 0$ and $f(.)$ is a concave non-decreasing function, is minimised by taking $k = k^*$ where k^* is the unique power of 2 satisfying

$$\frac{n}{3} \le k^* \;<\; \frac{2n}{3}$$

– the so-called *power of* 2 rule. Sinse (3) may be expressed in this form, it is seen that the power of 2 rule provides an optimum hierarchical tree for the additive screening case.

Furthermore, the power of 2 rule is optimum here regardless of the value of p, hence no prior knowledge is required of the probability of being defective is required to obtain the optimum algorithm (although independence and equal p is assumed).

11. The game of "mastermind" or "bulls and cows".

Consider the game known as "Mastermind" or "Bulls and Cows". The aim is to search for an unknown target T which is an ordered m-collection of some of d "colours" or "digits" which we numerate as $0, 1, \ldots, d-1$. The variant of the game we consider here is the "eastern" version in which none of the digits coincide. In other words, the target field is

$$\mathcal{T} = \{ T = \{t_1, \dots, t_m\} : t_i, t_j \in \mathcal{D} = \{0, \dots, d-1\},$$
$$t_i \neq t_j \text{ for any } i, j = 1, \dots, m, i \neq j\},$$
$$M = |\mathcal{T}| = d(d-1)\dots(d-m+1)$$

A test set $X = (x_1, \dots, x_m)$ is an arbitrary ordered m-collection of different digits from the same set $\mathcal{D}$, and so the test field coincides with the target field: $\mathcal{X} = \mathcal{T},\ R = M$. The test result $f(X, T)$ for given test set X and target T is the ordered pair (b, c) where b is the number of "bulls", i.e. such elements x_i of X that coincide with the corresponding elements t_i of T for $i = 1, \dots, m$, and c is the number of "cows",i.e. such elements x_i of X which correspond with some of t_j $(j = 1, \dots, m;\ j \neq i)$. In other words, for $X = (x_1, \dots, x_m),\ T = (t_1, \dots, t_m)$ we have $f(X, T) = (b, c)$ where

$$b = \sum_{i=1}^{m} 1_{[x_i - t_i]}, \qquad c = \sum_{i=1}^{m} 1_{[x_i \in \{t_1, \dots, t_m\} \setminus \{t_i\}]}.$$

If $m = 1$ then the mastermind search problem is trivial. Here we suppose $2 \leq m \leq d$.

Another version of the game, the "western version" allows t_i to coincide and the development is very similar.

Let us mention three simple properties of the search problem $\mathcal{J} = (\mathcal{T}, \mathcal{X}, f)$ determined by the variant eastern version.

Property 1: the test function $f(\cdot, \cdot)$ is symmetrical, i.e. for any $A_1, A_2 \in \mathcal{T} = \mathcal{X}$ the equality

$$f(A_1, A_2) = f(A_2, A_1)$$

holds.

Property 2: for any two sets $A_1, A_2 \in \mathcal{T}$ (or equivalently $A_1, A_2 \in \mathcal{X}$), there exists a transposition $\pi : \mathcal{D} \to \mathcal{D}$ of digits $0, 1, \dots, d-1$ such that $\pi(A_1) = A_2$ (this transformation may not be unique).

Property 3: if $\pi : \mathcal{D} \to \mathcal{D}$ is any transposition of digits and A_1 and A_2 are any elements of $\mathcal{T}$ (or $\mathcal{X}$) then

$$f(A_1, A_2) = f(\pi(A_1), \pi(A_2)),$$

and

$$f(A_1, \pi(A_2)) = f(\pi^{-1}(A_1), A_2). \tag{5}$$

Consider the non-sequential search problem. To study the existence of a γ-seperable non-sequential search algorithm with a given N apply Theorem 3. To do this we should have a method for computing $\sum_{j:j\neq i}(k_{ij})^n$ for various n and i where $k_{ij} = k(T_i, T_j)$, where $T_1, \dots, T_M$ are the elements of $\mathcal{T}$ ordered according to

a fixed rule, $k(G,B)$ is the number of $X \in \mathcal{X}$ such that $f(X,G) = f(X,B)$ for any $G, B \in \mathcal{T}$.

Proposition 5. For the above formulated search problem it holds that

$$\sum_{j=2}^{m}(k_{1j})^n \;=\; \sum_{j:j\neq i}(k_{ij})^n \tag{6}$$

for each n and $i = 1, \ldots, m$.

Proof. Let $i \in \{1, \ldots, m\}$ be fixed and $\pi_i : \mathcal{D} \to \mathcal{D}$ be a transposition of digits $0, \ldots, d-1$ such that $T_i = \pi_i(T_1)$. The existence of such a transposition follows from Property 2.

By definition, $k_{ij} = k(T_i, T_j)$ is the number of $l \in \{1, \ldots, M\}$ such that

$$f(T_l, T_i) \;=\; f(T_l, T_j). \tag{7}$$

According to Principle 3 we have

$$f(T_l, T_i) \;=\; f(T_l, \pi_i(T_1)) \;=\; f(\pi^{-1}(T_l), T_1),$$

$$f(T_l, T_j) \;=\; f(\pi^{-1}(T_l), \pi^{-1}(T_1)).$$

Denote $T'_j = \pi_i^{-1}(T_j)$ and $T'_l = \pi_i^{-1}(T_l)$. By definition of π_i we have $T'_i = T_1$ and

$$\{T'_j\}_j \;=\; \{T'_l\}_l \;=\; \{T_j\}_j \;=\; \mathcal{T}.$$

Thus, (7) is equivalent to

$$f(T'_l, T_1) \;=\; f(T'_l, T'_j)$$

and

$$k(T_i, T_j) \;=\; k(T_1, T'_j),$$

$$\sum_{j:j\neq i}(k_{ij})^n \;=\; \sum_{j:j\neq i}(k(T_i,T_j))^n$$

$$= \sum_{j:j\neq i}(k(T_1,T'_j))^n \;=\; \sum_{j:j\neq i}(k(T_1,T_j))^n \;=\; \sum_{j=2}^{M}(k_{ij})^n. \qquad \square$$

Equality (6) is followed by

$$\sum_{i=1}^{M}\sum_{j\neq i}\left(\frac{k_{ij}}{M}\right)^n \;=\; M\sum_{j=2}^{M}\left(\frac{K_{1j}}{M}\right)^n \tag{8}$$

and

$$\max_{i=1,\dots,M} \sum_{j\neq i} \left(\frac{k_{ij}}{M}\right)^n \;=\; \sum_{j=2}^{M} \left(\frac{k_{1j}}{M}\right)^n \tag{9}$$

where $M = d(d-1)\dots(d-m+1)$. These formulae simplify computations needed for applying Theorem 1. Moreover, in order to compute the right hand side of (8) and (9), it is neccesary only to compute all possible values of K_{1j} $(j = 2,\dots,M)$ and how many times they occur among $\{k_{1j}\}$. Denote M_1=number of times when $k_{1j} = k_l,\ j = 2,\dots,M;\ l = 1,\dots,L$. Then, evidently,

$$\sum_{j=2}^{M} \left(\frac{k_{1j}}{M}\right)^n \;=\; \sum_{l=1}^{L} M_l \left(\frac{k_l}{M}\right)^n. \tag{10}$$

□

Let

$$N^* \;=\; \min\Big\{n = 1,2,\dots \Big| \frac{M}{2} \sum_{l=1}^{L} M_l \Big(\frac{K_l}{M}\Big)^n \;\leq 1\Big\},$$

namely the sample size of a strongly separating non-sequential search algorithm, is given for some n and d. As an example consider the case $d = 5,\ m = 4$. Then the values of k_l for $l = 1,\dots,8$ are respectively 18, 20, 26, 32, 34, 36, 44, 46, and the corresponding values of M_l are 8, 12, 48, 24, 6, 10, 8, 3. The upper bound $N^* = 8$ in this case.

We complete the study of mastermind by give the results of three sequential algorithms. More extensive results will appear in a later paper. For all three algorithms the test sets are placed in lexigraphic order. For example for $m - 4,\ d = 8$ the order is 0123, 0124,...,4567.

Algorithm 1. "First consistent". At each iteration X_{t+1} is the first element from the order which is consistent with the data, i.e. the first element of $\mathcal{T}_t^*$.

Algorithm 2. "Best consistent". At each iteration we compare the entropy of the partitions generated by the elements of $\mathcal{T}_t^*$ and select the first which has maximum entropy.

Algorithm 3. "Best entropy". At each step we compare the entropies generated by *all* elements of $\mathcal{T}$ (consistent and inconsistent) and select the first having maximal entropy.

The results are given in Tables 3, 4 and 5. The length of the algorithms are given for every possible positions of the target producing frequencies and average length. The conclusion is that even the simplest algorithm is highly efficient. Algorithm 3 is always best or almost best among one-step-ahead procedures and seems close to optimal among all procedures. There is a slight uncertainty connected with placing the elements in standard order but this the does not significantly affect their efficiency.

	Length						
Algorithm	1	2	3	4	5	6	E(N(T))
1	1	12	73	161	81	8	3.991071
2	1	14	77	190	54	0	3.839286
3	1	14	100	213	8	0	3.633929

Table 3. Length frequency and average length of three sequential search algorithms for $m = 3,\ d = 8$.

	Length						
Algorithm	1	2	3	4	5	6	E(N(T))
1	1	15	208	791	547	118	4.322619
2	1	16	232	985	422	24	4.120833
3	1	10	327	1031	307	4	3.979167

Table 4. Length frequency and average length of three sequential search algorithms for $m = 4,\ d = 8$.

	Length								
Algorithm	1	2	3	4	5	6	7	8	E(N(T))
1	1	17	311	1949	3108	1227	105	2	4.823958
2	1	17	339	2657	3340	366	0	0	4.550000
3	1	15	484	2825	3231	164	0	0	4.452679

Table 5. Length frequency and average length of three sequential search algorithms for $m = 5, d = 8$.

12. Search for a subset of $\boldsymbol{T}$.

The problem where the researcher is interested in finding any subset of T rather

than the whole target also often occur. Algorithms searching for parts of T can be parts of many algorithms searching for T, they are natural in optimization problems (i.e. search for an optimal value of a function), when solving equations (i.e. search for a root of a function), etc. In some sense such problems are easier than the initial ones (since T is a subset of T) but they usually occur when the initial problems are too complicated or can not be solved.

Let us formalize new aims for the search problem $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$. We do not need any more achievability for any $T \in \mathcal{T}$ and so we shall not require solvability for $\mathcal{J}$ in this section. Instead of this we need some rules which permit us to decide whether we have found a subset of T or not. Let us consider three such rules which are natural for various problems. The first rule is evident:
Rule (i). If T is found than the problem is solved. Applying Rule (i) only we have the same problem as above.
Rule (ii). At each t-th iteration of a search algorithm we deduce for a set $Z \in \mathcal{B}(\mathbf{X})$ that $Z \subset T$ if $Z \subset T_i$ for each $T_i \in \mathcal{T}_t^*$ such that $\mu_t(T_i)) \neq 0$.

Note that the set Z in Rule (ii) cannot belong to $\mathcal{T}$ or $\mathcal{X}$. In most cases it is sufficient to consider Z in Rule (ii) one-point sets $\{x\},\ x \in \mathbf{X}$. Alternative situations may arise if we are interested in subsets of T of a given configuration.

So, to apply Rule (ii) in partice one should check whether there exists $Z \subset X$ such that $Z \subset T_i$ for each $T_i \in \mathcal{T}_t^*,\ \mu_t(T_i) > 0$. (Usually it is enough to check one-point sets Z only). Rule (ii) can easily be applied in some screening problems.

The third rule can be used for particular search problems which we shall call subtarget evaluable ones.
Rule (iii). After evaluating $f(X_t, T)$ for some $X_t \in \mathcal{X}$ we can check whether $X_t \subset T$ or not.

Definition 8. A search problem $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ is called a *subtarget evaluable* search problem if for any test set $X \in \mathcal{X}$ we can determine whether $X \subset T$ simultaneously with evaluating $f(X, T)$. The solution of equations with given accuracy, some optimization problems (problems with known optimal values), and screening in linear models without errors and equal known values of parameters corresponding to important factors give us examples of subtarget evaluable search problems. The main difference between rules (ii) and (iii) is that applying (ii) we check various subsets of $\mathbf{X}$ that cannot belong to $\mathcal{X}$ but applying (iii) we check only one subset of $\mathbf{X}$ belonging to $\mathcal{X}$. Sometimes all three rules can be applied.

Let us formulate a particular result concerning relative efficiency of deterministic search algorithms solving a subtarget evaluable search problem precisely and random γ-separating algorithms (i.e. algorithms that give a correct answer concerning a subset of T with probability not smaller than $1 - \gamma,\ \gamma > 0$).

Consider the search algorithms $\mathcal{A}$ that consist of two stages. In the first stage $N = N(\mathcal{A})$ test sets $X_1, \ldots, X_N$ are chosen from $\mathcal{X}$ according to some probability distributions

$$P_t(dX_t \,|\, X_1, \ldots, X_{t-1},\ f(X_1, T), \ldots, f(X_{t-1}, T)),\ t \geq 1.$$

In the second stage a decision is made according to Rule (iii). Denote by $\mathcal{R}_0$ the class of deterministic algorithms $\mathcal{A}$ (for them probability distributions P are degenerate) that solve the problem precisely for any $T \in \mathcal{T}$, and by $\mathcal{R}_\gamma$ the class of random algorithms solving the problem with probability not smaller than $1 - \gamma$. Denote

$$N_\gamma \ = \ \inf_{A \in \mathcal{R}_\gamma} N(A), \quad \gamma \geq 0. \tag{11}$$

Obviously, $\mathcal{R}_\prime \subset \mathcal{R}_\gamma$ and $N_\gamma \subset N_0$ for any $\gamma > 0$. The next assertion establishes other inequalities connecting N_0 and N_γ.

Proposition 6. Let $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ be a subset evaluable search problem, $\mathcal{X}$ consists of R elements and N_0 and N_γ are defined as in (11). Then for $\gamma \leq 1/e \approx 0.368$, it holds that

$$N_\gamma \ \leq \ N_0 \ \leq \ (N_\gamma + 1)\, \chi(\gamma, R) \tag{12}$$

where

$$\chi(\gamma, R) \ = \ 1 \ - \ [\log R \ + \ 1 \ - \ \log(1 - 1/\log\gamma)] \,/\, \gamma$$

The proof is omitted (see the Bibliography).

13. Root-finding and optimization.

Optimization and the solutions of equations, root-finding, present a rich class of search problems. They are often formulated with continuous rather than discrete sets. In this section we briefly consider them and also point to some differences between discrete and continuous search problems.

First consider search for solution of equations of the type

$$\psi(x) \ = \ 0, \quad x \in \mathbf{X}$$

where ψ is a continuous function from some functional class $\mathbf{\Psi}$, and $\mathbf{X}$ is a subset of $\mathbb{R}^n,\ n \geq 1$. Our considerations include the case where ψ is a vector-valued function and so $\psi(x) = 0$ is a system of non-linear equations.

The search problem $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ can be defined variously. The straightforward way is to define the test field as the family of all one-point sets $\{x\}$, i.e.

$$\mathcal{X} = \Big\{ \{x\},\ x \in \mathbf{X} \Big\}.$$

for target field $\mathcal{T} = \mathcal{B}(\mathbf{X})$, target T as a set of all solutions x^* of the equation $\psi(x) = 0$, i.e.

$$T = \{x^* \in X :\ \psi(x^*) = 0\}.$$

The test function f is naturally defined as

$$f(X,T) = \psi(x) \quad \text{for } X = \{x\},\ T \in \mathcal{T}.$$

Note that this test function f depends implicitly on $\mathcal{T}$.

Usually the problem is formulated so that there is no need to find the whole target set T, but only one point from it, i.e. only one solution of the equation $\psi(x) = 0$. Sometimes the functional class $\mathbf{\Psi}$ is defined in such a way that each function $\psi \in \mathbf{\Psi}$ has only one root in $\mathbf{X}$ and so search for

$$T = \{x^* \in X :\ \psi(x^*) = 0\}$$

is just the same as search for a subset of T. In this case the target field $\mathcal{T}$ coincides with $\mathcal{X}$ being the family of one-point sets. There is another, probably more natural, way of defining the target and target field. This method uses the fact that a computer solves numerical problems with some inaccuracy. Thus fix $\delta \geq 0$ and define

$$T_\delta = \{x \in X :\ \|\psi(x)\| \leq \delta\}$$

as a target set, $\mathcal{T} = \mathcal{B}(\mathbf{X})$, and formulate the search problem as the problem of finding some point x from T_δ. Note that this problem can be considered as a subtarget one (see Section 12) since for each test set $X = \{x\}$ we can conclude whether $X \subset T_\delta$ or not immediately after testing X.

Sometimes an alternative formulation of the same search problem is more instrictive: $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ where $\mathbf{X}$, $\mathcal{X}$ and f are as above, $\mathbf{X} \subset \mathbb{R}^n$, $\mathcal{X} = \{X = \{x\},\ x \in \mathbf{X}\}$, $f(X,T) = \psi(x)$, and $\mathcal{T}$ is the family of one point sets like $\mathcal{X}$, i.e.

$$\mathcal{T} = \{T = \{t\},\ t \in \mathbf{X}\}.$$

There are many targets in $\mathcal{T}$. Every $T = \{t\}$ corresponding to a point t from T_δ is a target but the problem is to search for these targets.

Direct application of entropy or Bayes considerations for construction of optimal or at least "good" algorithms often leads to very complicated intermediate calculations so that some where $\mathbf{X} \subset \mathbb{R}^n$, $\mathcal{X} = \{x\}$, $f(X,T) = \psi(x)$ for $X = \{x\}$, $T = \{t\}$ target $T = \{t\}$ is the one-point set corresponding to any point t from $T_\delta = \{x \in \mathbf{X} :\ \|\psi(x)\| \leq \delta\}$.

First suppouse that the function ψ can be evaluated without error. Let t iterations of a sequential algorithm be performed, $X_i = \{x\}$, $1 \leq i \leq t$, be the test sets (and x_i be the points where function ψ was evaluated),

$$\delta_i = \|\psi(x_i)\|, \quad \delta_t^* = \min_{1 \leq i \leq t} \delta_i.$$

If $\delta_t^* \leq \delta$ then the problem is solved since at least one of the points $x_1, \ldots, x_t$ belongs to T_δ. Suppose $\delta_t > \delta$ and consider

$$T_t^* = T_{\delta_t^*} = \{T = \{t\} \in \mathcal{T} :\ \|\psi(t)\| \leq \delta_t^*\}.$$

As above, $\mathcal{T}_t^*$ is the set of all $T \in \mathcal{T}$ which are consistent with the previous data (i.e. for every $T \in \mathcal{T}_t^*$ the supposition that T is a target does not contradict the previous test results).

In the $(t+1)th$ iteration, the test set X_{t+1} is selected from $\mathcal{X}$ depending on all previous test results or some of them:

$$X_{t+1} = \Phi_{t+1}(X_1, \ldots, X_t, f(X_1, T), \ldots, f(X_t, T)) \tag{13}$$

where Φ is some selection rule. (In some search algorithms several test sets are selected simultaneously, all of them except one are usually considered as auxillary) Naturally, $X_{t+1} = \{x_{t+1}\}$ is selected from $\mathcal{T}_t^*$. Such choice of X_{t+1} has two aims. First, there is a possibility that $x_{t+1} \in T_\delta$ and so X_{t+1} is a target. Secondly, this choice necessarily reduces $\mathcal{T}_t^*$ in the sence that $\mathcal{T}_{t+1}^* \subset \mathcal{T}_t^*,\ \mathcal{T}_{t+1}^* \neq \mathcal{T}_t^*$.

If $X_{t+1} \notin \mathcal{T}_t^*$ then the new observation $f(X_{t+1}, T)$ should not reduce $\mathcal{T}_t^*$ (in this case $\mathcal{T}_{t+1}^* = \mathcal{T}_t^*$) and it is useless from the one-step optimality point of view. This is precisely the nonpredictable case of Section 3. However, there are a lot of algorithms where such a choice of X_{t+1} is impossible. For example, sometimes it is very difficult to describe $\mathcal{T}_t^*$ and other principles are used for numerical solution of the equations (in particular, if equation $\psi(x) = 0$ is rewritten in the form $x = \tilde{\psi}(x)$ where $\tilde{\psi}$ is some other function then the algorithm $X_{t+1} = \{x_{t+1} = \tilde{\psi}(x_t)\}$ often converges to x^* and can be applied regardless of whether $X_{t+1} \in \mathcal{T}_t^*$ for each t).

The condition $X_{t+1} \in \mathcal{T}_t^*$, namely consistency for each t, is equivalent to the monotonicity of iterations

$$\|\psi(x_1)\| \geq \ldots \|\psi(x_t)\| \geq \ldots$$

This monotonicity property is highly desirable for algorithms of type (13) and holds for most of them. Assuming $X_{t+1} \in \mathcal{T}_t^*$, we can try to choose X_{t+1} optimally according to the above optimality criteria. This is sometimes possible when $\mathcal{T}_t^*$ can be described explicity.

As an example, consider one-dimensional algorithms for searching for the root x^* of a real valued continuous function ψ defined on an interval and having only one root. On each t-th ($t \geq 2$) iteration of the algorithms one has values $\psi(a_t)\,\psi(b_t)$ of the function at the boundary points of the interval $\mathcal{T}_t^* = [a_t, b_t]$ which is known to contain x^*. Also, it holds that $\psi(a_t)\,\psi(b_t) \leq 0$. Suppose this inequality is strict, since otherwise the problem is solved. The decision rule in the t-th iteration either terminates calculations, if a_t or b_t belongs to T_δ, or selects a new point $x_{t+1} \in (a_t, b_t)$ and sets

$$a_{t+1} = x_{t+1}, \quad bt + 1 = b_t$$

if $\psi(x_{t+1})\,\psi(b_t) \leq 0$ and

$$a_{t+1} = a_t, \quad b_{t+1} = b_t$$

otherwise. The problem now is how to choose x_{t+1} optimally. The solution to this problem depends on prior information concerning ψ, i.e. on the functional class $\mathbf{\Psi}$.

If the only information about ψ is that it is continuous and has exactly one root in $[a_t, b_t]$ then the middle point $x_{t+1} = (a_t + b_t)/2$ divides $\mathcal{T}_t^* = [a_t, b_t]$ by have and this division reduces the entropy of $\mathcal{T}_t^*$ optimally for the worst function $\psi \in \mathbf{\Psi}$. If in the interval $[a_t, b_t]$ the function ψ is well approximated by a linear function, i.e. $\psi(z) \approx c + dz$ for all $z \in [a_t, b_t]$ where c and d are some constants then the choice x_{t+1} as the intersection of the x-coordinate axis and the line passing through the points $(a_k, \psi(a_k))$ and $(b_k, \psi(b_k))$ is more natural. It can be shown that such a choice reduced the entropy of $\mathcal{T}_t^*$ optimally for the case where ψ is a trajectory of a differentiable random process intersecting the x-coordinate axis only once (in this case $\mathbf{\Psi}$ is a set of trajectories of a random process). Analogously a choice of x_{k+1} being the intersection of the x-axis and a quadratic function passing through $(a_k, \psi(a_k))$, $(b_k, \psi(b_k))$ can be interpreted for the case when ψ is well approximated by a second-order polynomial derivatives of ψ are evaluated simultaneously with values of ψ.

Consider briefly how the search methodology is modified for the case where evaluations of the function ψ are subjected to random errors (this case is the subject matter of stochastic approximation theory). The idea of selecting x_{t+1} among points corresponding to consistent one-point sets now gives nothing in its pure form because all subsets of $\mathcal{T}$ are consistent with probability one (for finite t any point of $\mathbf{X}$ may be the root of ψ because results are random variables). Nevertheless, the degree of consistency measured by the belief function μ_y is different for various test sets. So, a Bayes algorithm can be used or x_{t+1} can be chosen as the point for which $\mu_t\{\{x\}\}$ is maximal over $x \in \mathbf{X}$. In other words the test set is chosen which is the "most consistent" with the data. Morever, in each step consistency transforms in this case to consistency as an asymptotic property of the sequence $\{x_t\}$, $t \to \infty$, i.e. for suitable chosen decision rule for selecting x_t the convergence (say, in probability) $x_t \to x^* \in T_\delta$ should hold as $t \to \infty$. The Bayesian transformation of μ_t into μ_{t+1} is usually too complicated to be realised and simpler algorithms are often used in practice.

Consider now optimization search problems which can be formulated as problems of minimisation of a real-valued objective function ψ given on $\mathbf{X}$ and belonging to a functional class $\mathbf{\Psi}$. Denote by x^* any point in $\mathbf{X}$ such that $\psi(x) = \min_{x \in \mathbf{X}} \psi(x^*) = \psi^*$. Denote

$$T_\delta = \{x \in \mathbf{X} : |\psi(x) - \psi^*| \leq \delta\},$$

$$T'_\varepsilon(x^*) = \{x \in \mathbf{X} : \|x - x^*\| \leq \varepsilon\},$$

for any x^* such that $\psi(x^*) = \psi^*$ and $\delta > 0$, $\varepsilon > 0$. Parameters δ and ε correspond to an acceptable inaccuracy of the final decision with respect to function and argument values, respectively. Analogously to the root finding problems, optimization can be formulated as the search problem $\mathcal{J} = (\mathbf{X}, \mathcal{T}, \mathcal{X}, f)$ where $\mathcal{T} = \mathcal{X} = \{X = \{x\},\ x \in \mathbf{X}\}$, $f(X, T) = \psi(x)$ for any $X = \{x\} \in \mathbf{X}$, $T \in \mathcal{T}$, and the target is any one-point

set $T = \{z\}$ corresponding to $z \in T_\delta$ or $z \in T'_\varepsilon(x^*)$ for some x^* such that $\psi(x^*) = \psi^*$.

The minimal value ψ^* of a function ψ plays the same role as zero played in root-finding problems. If the value of ψ^* is known a-priori then these two classes of problem are very similar. But usually ψ^* is unknown which yields that for a given $x \in \mathbf{X}$ we cannot conclude whether x belongs to T_δ or not. For some functional classes $\mathbf{\Psi}$ like Lipschiz or convex ones it is possible to estimate accuracy of a current approximation to x^*. In these cases the precision of the final approximation of x^* is under control. In other cases we cannot say whether the problem has been solved with a given accuracy or not. Beliefs that the problem is solved precisely enough are based in these cases mainly on asymptotic properties of the algorithms or on estimates of the accuracy.

Consider briefly some principles which lie at the heart of optimization search algorithms. One of these is consistency: try to choose points at which to evaluate the objective function in those regions where values of this function are not greater than those we obtained earlier. In other words previous evaluations of the objective function do not contradict the supposition that any one of then is x^*. To formalize the consistency principle we should describe $\mathcal{T}_t^*$, the set of all consistent sets which contains all one-point sets $T = \{z\}$ that correspond to points z from the level set

$$\{z \in \mathbf{X} : \psi(z) \leq \min_{1 \leq i \leq t} \psi(x_i)\}. \tag{14}$$

As for the solution of equations case, consistency in optimization is the selection of the next test points is equivalent to the monotonicity of iterations, i.e. $\psi(x_1) \geq \psi(x_2) \geq \ldots \geq \psi(x_t) \geq \ldots$. This is easily achieved in the "local" optimization theory where the objective function is supposed to be uniextremal and the majority of algorithms are monotonic. The consistency principle is also realised for many global optimization methods: In particular all methods that can be regarded to the branch and bound methods use consistency ideas. Even some global random search algorithms are constructed so as to use the consistency principle. In these cases, consistency of subsets of $\mathbf{X}$ are tested statisticaly and so the inferences are statistical, i.e. valid with some probabilistic reliability.

Another basic principle in the construction of search algorithms is entropy or uncertainty reduction, i.e. those search algorithms are better which reduce the size of the level sets (14) faster. The well known Kiefer algorithm using Fibonnaci numbers presents an example of an optimal algorithm. It is optimal for the one-dimensional case when the sample size N is fixed, the functional class $\mathbf{\Psi}$ consist of all uniextremal functions given on an interval, and the criteria of optimality is the size of the level-set (14) for the worst case $\psi \in \mathbf{\Psi}$, after N steps of the algorithm. If N is not fixed and one-step reduction of the entropy for the worst one-dimensional function is used as the optimality criterion then the famous "golden section" is asymptotically optimal in a class of symmetric algorithms (symmetric algorithms choose the next point at intervals symmetrically placed with respect to the internal point of the interval where the previous observation was placed).

In multidimensional local optimization, fast reduction of the size of level sets is also the essense of a good algorithm. The problem of construction of N-step optimal (in entropy or risk function sense) algorithms is much more complicated than in the one-dimensional case. However, a lot of algorithms are known which asymptotically reduce the size of the level sets with a high speed.

In global optimization, the entropy or uncertainty reduction principle says that it is advantageous to select new test points in those subsets of $\mathbf{X}$ where either the reduction of size of the level set (14) can be expected or uncertainty concerning the objective function behavior is high. Immediate reduction of the level set sizes corresponds to local strategy and reduction of uncertainty about behavior of the objective function correspond to global strategy. Efficient global optimization algorithms combine both strategies. Applying a local strategy we can usually only find one or few local minimizers but not the global one. On the other hand, if local strategy is ignored then inefficient algorithms of the grid type would be generated.

Finally, likelihood is also fruitful in optimization techniques. In the construction of local optimization algorithms, it appears for instance in a Newton method when one logically approximates the objective function by a second order polynomial and the next point is selected in the direction of the minimizer of this polynomial. Note that asymptotic study of the Newton method and the some of its approximations (namely methods of conjugate directions and variable metrics) is one of the main points of local optimization theory.

There is a class of global random search methods known as generation methods which is popular in both theory and practice. Their essense is that in each t-th iteration a number of points are generated according to a probability distribution $\mathcal{P}_t$ which is constructed from the previous data in the same way as measures μ_t are constructed: they assign bigger measure to those subsets which are more consistent with the previous data. The reason for adding several observations taken from this distribution, rather than a single modal point is that there is uncertainty in the construction of $\mathcal{P}_t$, and this distribution is multimodal and it is unnatural to prefer one point to all others based only on this distribution.

Bibliography.

There are several thousand papers in the area of search in a wide variety of publications. In the non-stochastical literature these divide broadly into discrete search with foundation in logic and combinatorial theory and continuos search stemming originally from military applications. A recent book on the combinatorial theory is Ahlswede and Wegener (1987). This covers some classical Renyi theory, bifurcation, weighting and simple ideas in screening. Aigner (1988) has a similar coverge with additional sections on graph-theoretic techniques and coding theory. Classical results on computer search can be found in Knuth (1973,Vol. 3). Continuous problems are covered by Stone (1975) and the recent edition of Naval Research Logistics (Vol.

38, No. 3,1991) with an introduction by L.D.Stone.

The ideas in Section 3 are not found elsewhere with exactly the same results or in the same style. Similar ideas are, however, emerging in control theory, see Caines and Wong (1990) and in information and complexity, see Traub, Wasilkowski and Wozniakowski (1988).

The tree representation of Section 5 occurs throughout the combinatorial theory of search. The proof of the optimality of tree search is taken from Hu and Tucker (1971) and stems from earlier work of Huffman (1952).

The use of belief measures and Bayes (Section 6 and 9) is drawn from standard decision theoretic statistics. Within AI there are sequential methods which are referred to variously as discrete decision processes, dynamic decision processes, heuristic search etc. A recent collections are Kanal and Kumar (1988).

Entropy is perhaps the most widely used portmanteau probabilistic tool. Martin and England (1981) is comprehensive. Maximum entropy methods are used in spatial sampling (Shewry and Wynn,1987, Sacks et al.,1990) and in screening and group testing (Sobel and Groll, 1959, Mitchell and Scott, 1987). The first definitive statements of the group-testing are Dorfman (1943) and Watson (1961). The solution of the optimum binary group-testing problem using alphabetic trees is by Hwang (1976). Additive procedures were introduced by Pfeifer and Enis (1978) and the optimum hierarchical procedure was given by Hwang, Pfeifer and Enis (1981). Proof that the power-of-two rule gives the optimum tree for solving recursive equations of the form (4) was obtained by Glassey and Karp (1976).

The original papers by Renyi are Renyi (1965,1969). This work has been extensively developed by M. Maljutov and other Russian scientists. Some of it appears in Ermakov (1983,ed) and is extended in the papers by the present authors (O'Geran, Wynn and Zhigljavsky,1991 b,c). These paper continue the use of Mastermind as a test case. References to this game appear also in books on computer games e.g. Arsac (1985).

Zhigljavsky (1991) contains many different algorithms for global optimization of which Sections 12 and 13 are a synthesis drawing on the style of other sections of the present paper. There are numerous works on "local" optimization for example Dennis and Schnabel (1983) and Fletcher (1980). The proof of Proposition 6 follows the lines of Zhigljavsky (1991, Section 6.2.4).

Acknowledgement.

This research was conducted within the Engineering Design and Quality Centre at City University. The authours acknowledge the support of the Science and Engineering Research Council, UK.

References

[1] Ahlswede R., Wegener I. (1987). *Search Problems*. Wiley, New York.

[2] Aigner M. (1988). *Combinatorial Search*. Wiley, New York.

[3] Arsac J.D. (1985). *Jeux et Casse-tête. A Programmer*. Bordas, Paris.

[4] Caines P., Wang S. (1989). Classical and logic-based regulator design and its complexity for partially observed automata. *Proc. 28-th IEEE Conf. Dec. and Control*. Tampa, Florida, 132-137.

[5] Dennis J.E., Schnabel R.B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey.

[6] Dorfman R. (1943). The detection of defective members of large populations. *Annals of Mathematical Statistics*, 14, 436-440.

[7] Ermakov S.M., ed. (1983). *Mathematical Theory of Experimental Design*. Nauka, Moscow. (in Russian)

[8] Fletcher R. (1980). *Practical Methods of optimization*. Wiley, New York.

[9] Glassey C.R. and Karp R.M. (1976). On the optimality of Huffman trees. *Siam Journal on Applied Mathematics*, 31, 368-378.

[10] Hu T.C., Tucker A.C. (1971). Optimum computer search trees and variable-length alphabetic codes. *SIAM Journal on Applied Mathematics*, 21, 514-532.

[11] Huffman D.A. (1952). A method for the construction of minimum redundancy codes. *Proc I.R.E.,* 40, 1098-1101.

[12] Hwang F.K. (1976). An optimum nested procedure in binomial group testing. *Biometrics*, 32, 939-942.

[13] Hwang F.K., Pfiefer C.G., Enis P. (1981). An optimal hierarchical procedure for a modified binomial group-testing problem. *Journal of the American Statistical Association*, 76, 947-949.

[14] Kanal L., Kumar V. (1988). *Search in Artificial Intelligence*. Springer-Verlag, New York.

[15] Knuth D. (1973). *The Art of Computer Programming,vol 3*. Addison-Wesley, London.

[16] Martin N.F.G., England J.W. (1981). *Mathematical Theory of Entropy*. Addison-Wesley, London.

[17] Mitchell T.J., Scott D.S. (1987). A computer program for the design of group testing experiments. *Communications in Statistics. Theory and Methods*, 16, 2943-2955.

[18] O'Geran J.H., Wynn H.P., Zhigljavsky A.A. (1991a). Renyi-type randomization theory for search length upper bounds. *Stochastic Optimization & Design* (submitted).

[19] O'Geran J.H., Wynn H.P., Zhigljavsky A.A. (1991b). Mastermind as a test-bed for search algorithms. *American Mathematical Monthly* (submitted).

[20] Pfiefer C.G. and Enis P. (1978). Dorfman-type group testing for a modified binomial model. *Journal of the American Statistical Association*. 73, 588-592.

[21] Pierce C.P. (1901). In Charles Hartshorne, Paul Weiss and Arthur Banks (eds). *The collected Papers of Charles Saunders Peirce*, 8 vols. Harvard University Press, Cambridge, Mass.

[22] Renyi a. (1965). On the theory of random search. *Bulletin of the American Mathematical Society*, 71, 809-828.

[23] Renyi A. (1969). *Lectures on the Theory of Search*. University of North California, Mimeo Series.

[24] Sacks J., Welch W.J., Mitchell T.J., and Wynn H.P. (1989). Design and analysis of computer experiments. *Statistical Science*, 4, 409-435.

[25] Shewry M.C., and Wynn H.P. (1987). Maximum entropy sampling. *Journal of Applied Statistics*, 14, 165-170.

[26] Sobel M., and Groll P.A. (1959). Group testing to eliminate efficiently all defectives in a binomial sample. *Bell System Technical Journal*, 38, 1179-1252.

[27] Stone L.D. (1975). *Theory of Optimal Search*. Academic Press, N.Y.e.a.

[28] Traub J.F., Wasilkovski G.W., and Wozniakowski H. (1988). *Information-Based Complexity*. Academic Press, New York.

[29] Watson G.S. (1961). A study of the group screening method. *Technometrics,* 3, 371-388.

[30] Zhigljavsky A.A. (1991), *Theory of Global Random Search*. Kluwer Academic Publishers, Dordrecht.