Omega 38 (2010) 12 – 19

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Sequencing questions to ferret out terrorists: Models and heuristics☆

P.S. Sundararaghavan, Anand Kunnathur, Xiao Fang*

Information, Operations and Technology Management Department, University of Toledo, Toledo, OH 43606, USA

ARTICLE INFO

Article history:
Received 14 March 2008
Accepted 30 January 2009
Available online 21 February 2009

Keywords:
Information security
Decision support systems
Information systems

ABSTRACT

Consider the problem of granting boarding clearance for a large population of air travelers while ferreting out any potential terrorists and denying them entry, in a short time. It is assumed that the probability of having a terrorist in the group is very small. Further assume that the process consists of asking a series of questions and the decision to clear or deny is dependent on the answer set. An efficient sequencing of the questions may reduce the number of questions needed to be asked in order to reach a decision. This problem is modeled as a question sequencing problem. The problem is intrinsically hard and hence we develop two approaches to solving the problem. The first uses a traditional greedy heuristic approach exploiting the relationship between answers and the outcome. The second adopts the decision tree approach used in classification problems to this problem. We also report on the performance of the two heuristics which does exceptionally well on problems with a very low probability of occurrence.

1. Introduction

After the 9–11 attack, the free world in general, and the USA in particular, have been catapulted into a new era, where security has taken precedence over customer convenience and even individual rights in some arenas. The problem discussed in this paper has many potential applications in passenger screening, cargo screening, port security, quality control among others. The arena we specifically focus on in this work deals with passenger screening prior to boarding at airport security checkpoints. Specifically, we address the problem of optimizing the time required to perform security checks of passengers seeking to board a given flight. The airport security problem has special features that are not found in many other search problems. For example, there is a limited time to clear all passengers for boarding so that flight is on time. As practiced now at most US airports, any delay in one passenger's search delays all passengers' departure. However, any one passenger may be denied boarding without affecting the status of all others that are cleared for a particular flight. The cost of denying boarding is small compared to the cost of allowing boarding of a terrorist. At the same time the delays in passenger screening are a significant factor in reducing passenger traffic, especially on short-haul flights. The number of potential terrorists is small compared to the number of passengers, and airport screening is somewhat like looking for a needle in a haystack. There are two somewhat similar problems, which have received some research attention over the years. First is the problem of searching for drugs in the US highway system, customs, and in international waters. Here again, the search is done one at a time and there is a possibility of searching the "wrong" person. Quite often, the "suspect" is taken out of the waiting line and searched. Thus, the time taken to search one person has no effect on any other person. The number of drug smugglers is potentially a larger fraction of the population searched compared to the terrorist population as a fraction of the flying public. However, cost of a failed drug search is relatively small, in contrast with the airport security problem. A problem with similar overtones is the question sequencing problem in sampling. However, the sample is tied to the population irrevocably, and unlike the airline search problem, the whole lot is accepted or rejected. Another problem that is related is the problem of designing an optimal length test so as to estimate whether the knowledge level of the test taker exceeds the minimum to obtain a pass [20]. Section 2 surveys the current research on related problems. Section 3 presents some definitions and notations. In Section 4, we present an exact model of a restricted version of the airline security problem and present some solution procedures. In Section 5, we discuss a more general version of the model and present some heuristics and ideas for other approaches to solve the problem. Section 6 contains concluding remarks and avenues of future research.

☆This manuscript was processed by Associate Editor Teo.

* Corresponding author. Fax: +1 419 530 2290.
E-mail addresses: psundar@utnet.utoledo.edu, akunnat@utnet.utoledo.edu, xiao.fang@utoledo.edu (X. Fang).

0305-0483/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.omega.2009.01.002

2. Literature search

The work presented in this paper falls under the broad and emerging category called research on Intelligence and Security Informatics (ISI). According to [3], ISI is "the study of the use and development of advanced information technologies, systems, algorithms, and databases for national security related applications through an integrated technological, organizational, and policy based approach". Prior research on ISI includes applications of such advanced techniques as information sharing and collaboration, data mining, text mining and social network analysis to critical national security areas such as intelligence and warning, domestic counter-terrorism and border and transportation security [15]. ISI development in the area of intelligence and warning primarily focuses on the detection of potential terrorist activities [3]. Wang et al. [22] developed automatic methods for detecting deceptive identity records in police data. In [24], a principle clusters analysis approach for analyzing user navigations on the Web was presented. The approach can be used to identify prominent navigation clusters on different topics including terrorism related topics. ISI applications in the area of domestic counter-terrorism aim at improving information sharing and crime analysis abilities of local, state, and federal crime fighters [3]. One example of ISI application in this area is COPLINK Detect, an intelligence system that automatically extracts relationship information from crime incident data [9]. An outlier-based approach to resolve the criminal incident association problem [12] is a recent ISI application in this area. This paper is particularly related to the ISI research in the area of border and transportation security, which employs information technologies to create secure borders and transportation systems. Prior research in this area includes information sharing among government agencies, criminal activity network analysis, and anomaly detection in real-time data streams among others. Marshall et al. [13] proposed a framework for integrating data stored in different law enforcement agencies. Criminal activity networks were then generated based on the integrated data to identify potential associations among criminal incidents. Warner et al. [23] designed a secure data interoperation model for integrating data from different government agencies, ports, and customers divisions. The integrated data can be used to detect anomalies such as non-conforming shipments, abnormal behavior of inbound cargos among others [1]. All of the forgoing research gives us good data sources to formulate the problem discussed in this paper. As will be seen later, the formulation requires prior probabilities for potential terrorist incidence in a population. This research studies transportation security. In particular, we address the problem of optimizing the time required to perform security checks of passengers seeking to board a given flight.

Our specific research, namely, optimal sequencing of questions does not appear to have received any attention in the research literature. There are statistically oriented analyses for test construction as in Wainier and Wang [21] reporting on the TOEFL test for foreign students, and similar analysis for the GRE exam [18] which appear peripherally related to our topic area. The information-theoretic sequence alignment problem appears to be related to the problem at hand. Allison [2] reports on the development of algorithms for aligning sequences. The sequence alignment problems is related more to the generation of a complete set of pass or fail constraints, called edits, in our problem and less to the optimal sequencing of the questions whose responses will be screened through the edit system developed for the situation. The problem of edit generation, given true/false type of answers to posed questions, is closely related to the satisfiability problem [7]. The development of an optimal sequence of questions whose responses when processed through the edits result in a definitive pass or fail of the edit system requires identification of those questions (clauses in the satisfiability problem) which are the most critical to ask early on. In that sense the question sequencing problem appears related to job sequencing problems in scheduling [7]. Since the probability of a response being labeled trustworthy or not is dependent on what previous questions have been asked, our problem is also related to the maximum likelihood problem of Bayesian statistics.

Considering the complexities of the underlying problems and their computational intractability–satisfiability, optimal sequencing with constraints to be satisfied and likelihoods to be revised—the problem at hand is unlikely to be solvable to optimality with acceptable computational expenditures. We focus on development of heuristic algorithms in an attempt to present viable, dynamically usable mechanisms for screening subjects for making accept/reject decision.

3. Problem description and motivation

Suppose that we have a system charged with checking any item for suitability for a particular use. Each item is subjected to a series of k tests. Each test has an n-nary outcome. After conducting all the k tests, one can find out whether the item is suitable for the particular use or not. Since the final outcome is binary (found the item to be fit for use or not), there will be many overlaps in the series of test results that lead to the same final result. By suitably arranging the sequence of the tests, one would be able to make the final conclusion without actually conducting all the tests. Hence, it may be useful to sequence the tests so as to minimize the number of tests to be conducted in order to reach the final conclusion. This may be formulated as a question sequencing problem. Let us look at specific versions of such a problem with an application in air passenger clearance.

Consider an airline passenger processing system, where each passenger is asked a series of questions which have binary (yes/no) answers, to find out whether there is any risk in clearing the passenger for travel. At the end of the interview the passenger is either cleared or denied entry to the plane. This system can be posed as a problem of the type described above. Next, we present a formal definition for such a problem.

Passenger Screening Problem (*PSP*): Let $\eta = \{1,2,\ldots,n\}$ be a set of questions and $\alpha = \{0(\text{No}),1(\text{Yes})\}$ be a set of answers. All answer sequences to the n questions are mapped to one of two outcomes: "Clear for travel" and "Denied travel". Given a typical passenger (flyer) who is going through security check and given $p_{ij} i \in \eta$ and $j \in \alpha$, which is the probability of answering question i with answer j, what should be the order in which the questions must be posed that would minimize the expected number of questions to be asked of a typical passenger.

Next we present an example to illustrate the issues involved in this problem.

Example 1. Suppose that there are two questions A and B and the attendant details as described below. The objective of the problems is to find the correct order to ask the questions in order to minimize the expected number of questions asked (Tables 1 and 2).

It is easily seen that if the answer to Question A is "Yes", the outcome is always C, regardless of the answer to Question B. Hence, there is no need to ask Question B in that case. If the answer to Question A is "No", then one has to ask Question B in order to complete

Table 1
Probability distribution for answers to questions.

	Question A	Question B
P(Yes)	0.7	0.9
P(No)	0.3	0.1

Table 2
Permutation of answers and corresponding outcomes.

A	B	Outcome (Clear (C) and Deny (D))
Yes	Yes	C
Yes	No	C
No	Yes	D
No	No	C

Table 3
Question set Q and the p_{ij} matrix.

Question	Prob. of yesYes (coded as 1)	Prob. of No (coded as 0)
A	0.2	0.8
B	0.3	0.7
C	0.1	0.9

the check. For the question order AB, if the first answer is "Yes", the number of questions to be asked is 1 and if the first answer is "No", the number of questions to be asked is 2, which leads to the following expression.

The expected number of questions to be asked (for order AB) $= 0.7\times1+0.3\times2 = 1.3$.

If the question order is BA, by similar logic, one can calculate the expected number of questions to be asked as follows.

The expected number of questions to be asked (for order BA) $= 0.1\times1+0.9\times2 = 1.9$.

Appropriate sequencing in this small example saves around 30%. This problem may appear to be solvable by using the well-known Karnaugh map technique, which is used to minimize logical circuitry, also called Boolean minimization. In logical circuitry, the problem is, given a Boolean logical operation between two Boolean variables, how many distinct functions can be represented by a given set of switches. In the PSP problem, there is no Boolean logic used to combine the answers to individual questions in order to derive the outcome (clear or deny). The relationship between the answer table and the outcome (there are only two outcomes (clear or deny)) is totally arbitrary in the PSP problem. Further, as noted in [5], Karnaugh map is most useful for 2–4 logical variables and becomes steadily unwieldy beyond those numbers whereas we attempt problems up to size 20. Hence, we did not explore this tool further. Next we propose a formal model to address PSP.

4. Definitions and solution procedures

Consider the Passenger Screening Problem as described in Section 3 with n (yes/no) questions, and an attached final decision based on the answers to the n questions. That is, all the permutation of answers is uniquely mapped to a binary set of outcomes denoted as "Clear" or "Deny". Further, for each question, the probability of "yes" or "no" is assumed to be known. The problem is to find the order in which questions should be posed in order to minimize the expected number of questions to be asked.

First, some definitions are given to characterize the answer set for a given order in which questions are posed.

Primitive Edit: Any unique combination of answers for the n questions in the order in which they are posed.

Probability of a primitive edit: It is the joint probability of obtaining the sequence of answers represented by the primitive edit.

Clear Set (C): The set of all primitive edits that are mapped to the outcome denoted as "Clear".

Deny Set (D): The set of all primitive edits that are mapped to the outcome denoted as "Deny".

Answer Set (A): The set of all 2^n unique answer sequences (set of all primitive edits) for the given set of n questions and the associated order in which they are posed.

Assumption 1. $C \cup D = A$ and $C \cap D = \Phi$.

We present a 3-question numerical example which is used throughout to illustrate various solution procedures.

Example 2. Suppose that there are three questions A,B, and C with the probabilities given in Table 3, primitive edits and the corresponding outcomes (C and D) given in Table 4. The objective is still to find the sequence that minimizes the expected number of questions to be asked. It can be easily seen that set C contains primitive edits 1, 3, 4, 7, and 8 and set D contains all the rest of the primitive edits. Find the order in which to pose the questions so as to minimize the expected number of questions asked.

4.1. Solution approach using complete enumeration

There are 3! ways in which the questions may be asked and the optimal solution is characterized by the solution which requires the smallest expected number of questions to be asked. For illustration, question order ABC, along with all possible answers, and the corresponding expected number of questions to be asked are given in Table 5. Suppose the answer sequence is (0, 1, 0). It may be noted from primitive edits 3 and 4 produce an outcome of C, meaning clearance. However, it is not necessary to ask question C, since an answer of 0 to question A, and 1 to question B lead to clearance regardless of the answer to question C. Hence the number of questions to be asked is 2 for answer combination 0 1 for question A and B, respectively. The probability of such an answer sequence is $0.8\times0.3 = 0.24$. Finally the expected number of questions to be asked is $0.24\times2 = 0.48$. For the primitive edit (0 1 1) the number of questions to be asked is 0, since it is already taken into account by the primitive edit (0, 1, 0). Summarized answers are given for all the 3! orderings in Table 6. The optimal solution is to ask the questions in the order: BAC, which requires an average of 2.26 questions.

4.2. Heuristic solutions

Clearly, solving by complete enumeration is not viable for problems of large or even medium size. A feasible approach may be through heuristic methods. If a question becomes redundant based on answers to previous questions, it is clear that such a question should be posed as late as possible to take advantage of the redundancy. In Example 2, question C, for the question order ABC, is redundant if the answers to A and B are 0 and 1, respectively. Hence identifying such redundancies systematically and exploiting those in solution development is the basis for the heuristic to be presented later. First, we introduce some additional definitions.

Iso-edit: Two primitive edits i and j are said to be a pair of iso-edits of order 1 (i,j), if both $i,j \in$ C or both $i,j \in$ D and they have identical answers for any of the $(n-1)$ questions out of the set of n questions. For example, edits 1 and 3 of Table 5, with outcome C is a pair of iso-edits of order 1. By extension, a set of 2^k primitive edits are said to be iso-edits of order k whenever all of them belong to either C or D, and they have identical answers for any $(n–k)$ questions. Thus, the 2^k primitive edits exhaust all combinations of answers for the remaining k questions (notice that identical answers to $(n–k)$ questions implies the latter).

General edits: A pair of iso-edits (i,j) of order 1 can be combined to form a general edit (i,j) of order 1. For example, edits 1 and 3 of Table 5 can be combined into the general edit 1 shown in Table 9. Further, it is equivalent to a primitive edit with all the answers unchanged except that the answer for the single question

Table 4
Primitive edit set with C and D denoted against primitive edits.

Edit #	Question A	Question B	Question C	Designated membership in C or D	Probability of edit
1	0	0	0	C	0.504
2	0	0	1	D	0.056
3	0	1	0	C	0.216
4	0	1	1	C	0.024
5	1	0	0	D	0.126
6	1	0	1	D	0.014
7	1	1	0	C	0.054
8	1	1	1	C	0.006

Table 5
Detailed analysis for question sequence ABC.

Primitive edit #	Question			Probability of the answer to A	Probability of the answer to B	Probability of the answer to C	Joint prob.	#of Qns required to make a decision	Decisions (clear: C and deny: D)	Exp.# of questions
	A	B	C							
1	0	0	0	0.8	0.7	0.9	0.504	3	C	1.512
2	0	0	1	0.8	0.7	0.1	0.056	3	D	0.168
3	0	1	0	0.8	0.3	0.9	0.24[a]	2	C	0.48
4	0	1	1	0.8	0.3	0.1	–	0[b]	C	0
5	1	0	0	0.2	0.7	0.9	0.14[a]	2	D	0.28
6	1	0	1	0.2	0.7	0.1	–	0[b]	D	0
7	1	1	0	0.2	0.3	0.9	0.06[a]	2	C	0.12
8	1	1	1	0.2	0.3	0.1	–	0[b]	C	0
										2.56

[a]In these cases question C need not be asked and hence the probability is the product of P(answer /A) $*$ P(answer /B).
[b]In these cases the previous edit has already made the decision and hence this primitive edit will never be used.

with differing answers replaced by the letter C, if both $i,j \in$ C[1] and replaced by D if both $i,j \in$ D. A general edit of order k can be developed from a set of iso-edits of order K. Such an order K general edit is similar to a single primitive edit with n–k identical responses, and with all differing answers for the k questions replaced by C or D depending upon which set the primitive edits belong to.

Union of general edits: It is the set of all general edits of order $k \geqslant 1$.

Generating general edits: General edits are crucial part of the solution process. Lemma 1 defines the characteristics of general edits of order 1.

Let n be the number of questions. The number the primitive edits corresponding to n questions will be equal to 2^n. All general edits of order 1 that may be formed from the primitive edits can be represented by a $2^n \times 2^n$ general edit of order 1 incidence matrix. The rows and columns of the incidence matrix correspond to the primitive edits. An entry M in cell (i,j)[2] means no general edit of order 1 exists between the primitive edits i and j. A positive integer p in cell (i,j) means that there exists a general edit of order 1 between primitive edit i and j,[3] with entries differing only in position p. Such a matrix will be symmetric and hence only the upper diagonal elements may need to be filled. Table 7 illustrates this for case of $n = 3$. Lemma 1 will help us to develop a formal approach to deriving a table such as this for any problem size.

Lemma 1. *There exists a pattern of combinations for generating general edits from a, suitably arranged, exhaustive set of primitive edits.*

Proof. Table 7 is an illustration of Lemma 1 for $n = 3$. The lemma is fairly self evident if we look at the pattern of primitive edits and observe in which position a given pair differs. Hence, it is left to the reader. ■

Table 6
Evaluation of all possible solutions.

Question order	Expected number of questions required to make a decision
ABC	2.56
ACB	2.28
BAC	2.26
BCA	2.33
CAB	2.28
CBA	2.63

Table 7
Incidence matrix of general edits of order 1 for $n = 3$, assuming that all edits are in set C (Clear).

Row/Columncolumn label	1	2	3	4	5	6	7	8
1	–	3	2	M	1	M	M	M
2		–	M	2	M	1	M	M
3			–	3	M	M	1	M
4				–	M	M	M	1
5					–	3	2	M
6						–	M	2
7							–	3
8								–

Example 3. Apply Lemma 1 for a problem with $n = 3$ to generate the incidence matrix assuming all primitive edits are either in C or in D.

Number the primitive edits as 1–8 as shown in Table 5. This is one way to suitably number an exhaustive set of primitive edits as described in Lemma 1. Refer to row 1 of Table 7 as you read the explanation that follows. As per the patterns of combination described in Lemma 1, primitive edit 1 can only combine with primitive edits 2, $2+2^0 = 3$, and $2+2^0+2^1 = 5$. Further the position at which it

[1] For example, 110 and 111 (primitive edits 7 and 8 in Table 5) are replaced by 11C.
[2] For example, 111 and 001 (primitive edits 8 and 2 in Table 5).
[3] For example, 110 and 111 (primitive edits 7 and 8 in Table 5).

Table 8
Incidence matrix of general edits of order 1 for $n = 3$ (using data of Table 4).

Row/Column label	1	2	3	4	5	6	7	8
1	–	M	2	M	M	M	M	M
2		–	M	M	M	1	M	M
3			–	3	M	M	1	M
4				–	M	M	M	1
5					–	3	M	M
6						–	M	M
7							–	3
8								–

Table 9
Union of general edits.

General edit #	Question A	Question B	Question C	Primitive edits used
1	0	C	0	1 and 3
2	1	1	C	7 and 8
3	0	1	C	3 and 4
4	C	1	0	3 and 7
5	C	1	1	4 and 8
6	C	1	C	3, 4, 7, and 8
7	1	0	D	5 and 6
8	D	0	1	2 and 6

can combine with edit 2 (this is in the 2^0th (or first) super diagonal), is given by n which is 3 in this example. Combination with edit 3, which is in the 2^1st (second) super diagonal, can only be done at position $(n-1) = (3-1) = 2$. Finally, combination with edit 5, which is in the 2^2 (fourth) super diagonal, can only be done at position $(n-2) = (3-2) = 1$. Combination of primitive edit 2,3,…,8 can be similarly derived using Lemma 1. Table 7 gives the complete results.

Lemma 2. *Let n be the number of questions and let 2^n be the corresponding primitive edits. All general edits of order k may be formed from these edits using a pattern similar to Lemma* 1.

Suppose we impose the restrictions of Example 2 as shown in Table 4 on the matrix of Table 7, we get Table 8. For example Cell(1,2) is changed from 3 to M, because edit 1 and 2 are in different sets (1 in C and 2 in D). Similarly all other non-M entries are checked against the data and suitably changed, thus reducing the computation further.

Lemmas and general edit generation: The application of these lemmas will greatly reduce the computational burden in generating general edits. For any problem size, for each edit, there is at most n other edits to be considered for combing into a general edit. The list of candidate edits is deterministic and easily generated. The candidates are further pruned by applying the condition that the two edits should be in the same set. Thus, these two lemmas allow us to conceptualize heuristics exploiting the fact that it makes sense to worry about the order of the questions only up to the point where ordering affects the efficiency of the process. Suppose in a problem instance, there are n questions and the maximum order of general edits that can be generated is k, then, it is clear that $(n–k)$ questions can be ordered in any fashion without affecting the efficiency of the solution. We still need to pick the "right" set of k questions and order them as best as possible in the last k positions and place the remaining questions in any order. Heuristics 1 and 1a use these principles to build reasonable solutions.

Heuristic 1. *Step* 1: Generate the union of general edits using the information in Table 8 (or similarly created table for the problem at hand).

Step 2: Choose the question with maximum number of C's or D's from the table of union of general edits. Break ties arbitrarily. Let this be $q_{[n]}$, the question picked for position n.

Step 3: For choosing $q_{[n-i]}$, question for position $(n-i)$, among the remaining questions, find that question which has maximum number of C's or D's counting only general edits of order $(i+1)$ and only those edits which already has C's and D's in the questions already chosen for positions (n), $(n-1),\ldots,(n-i+1)$. Break ties arbitrarily. Place this to be question $q_{[n-i]}$. If at any time all questions have no C's or D's satisfying the condition that they are in general edits of the correct order and that those edits have C's and D's in all the questions already chosen, then the remaining questions may placed in any order.

Let us use Example 2 to illustrate Heuristic 1 in a stepwise process.

Step 1: Generate a union of general edits as shown in Table 9.

Step 2: There are two candidates, question A and C with maximum number of C's or D's in the Table 9. Breaking ties arbitrarily, one can choose Question C to be the question in position 3.

Step 3: Question A has a general edit of order 2, which has a C in the same general edit as Question C. There are no other general edit of order 2, satisfying this condition for Question A or Question B. Hence, pick Question A for position 2. That automatically assigns Question B to position 1. Thus, the order is BAC. This also happens to be the optimal solution for Example 2.

The computational burden of Heuristic 1 is fairly heavy, thus limiting the size of the problems that may be solved. Heuristic 1a uses the probability of occurrence to combine the general edits, but once the higher order edit has been generated, the lower edits are discarded. This dramatically reduces the computational burden, but does impede the effectiveness. This also allows us to solve considerably larger sized problems.

Heuristic 1a. *Step* 1: List all primitive edits in descending order of their probabilities.

Step 2: Generate the non-overlapping union of general edits starting with the primitive edit with the highest probability. In this method, if two general edits of order i could be combined into a general edit of order $(i+1)$, then the edit of order $(i+1)$ is retained in the set while dropping the two edits of order i.

Step 3: Choose the question with maximum number of C's or D's from the table of union of general edits. Break ties arbitrarily. Let this be $q_{[n]}$, the question picked for position n.

Step 4: For choosing $q_{[n-i]}$, question for position $(n-i)$, among the remaining questions, find that question which has maximum number of C's or D's counting only general edits of order $(i+1)$ and only those edits which already has C's and D's in the questions already chosen for positions (n), $(n-1),\ldots,(n-i+1)$. Break ties arbitrarily. Place this to be question $q_{[n-i]}$. If at any time all questions have no C's or D's satisfying the condition that they are in general edits of the correct order and that those edits have C's and D's in all the questions already chosen, then the remaining questions may placed in any order.

Heuristic 2. Solution to the Passenger Screening Problem can be obtained by learning a classification model from a set of primitive edits and their associated outcomes. The learned model is then used to classify future passengers as either clear or deny. Hence, traditional classification methods such as decision tree induction algorithms can be adapted to solve PSP. A decision tree is a classification model with tree structure, where each internal node represents a test on an attribute, each branch indicates a test outcome and each leaf node represents a class. Algorithm ID3 [16] is a greedy decision tree induction algorithm that constructs a decision tree recursively from top to down. It employs entropy to quantify uncertainty in a set and picks the attribute with the greatest entropy reduction as the test attribute at each internal node in a decision tree. Algorithm C4.5 [17] enhanced the performance of ID3 in such areas as decision tree pruning and handling continuous-value attributes. Efficient and scalable decision tree induction algorithms were proposed in [8,14,19].

By mapping the Passenger Screening Problem to a classification problem, questions in a primitive edit are synonymous to description

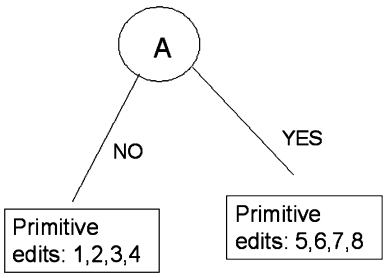

Fig. 1. A partially inducted decision tree.

attributes in a classification problem. The outcome associated with a primitive edit can be treated as a class label attribute. A decision tree is constructed through a repetitive process with the following steps: choosing an appropriate description attribute (i.e., question) with the greatest entropy reduction, testing on the chosen description attribute, and creating branches based on testing outcomes (i.e., answers to a question). The process stops when all leaf nodes in the decision tree consist of primitive edits associated with a single outcome (i.e., Deny or Clear). Let us use Example 2 to illustrate Heuristic 2 in a stepwise process.

Step 1: Choose an appropriate description attribute (i.e., question) with the greatest entropy reduction.

According to [16], the entropy of the overall set of primitive edits before partitioned by any question can be calculated as shown below.

$$E(\text{overall}) = -P(C) * \log_2^{P(C)} - P(D) * \log_2^{P(D)} = 0.71,$$

where $P(C) = 0.804$ is the probability of clear in the overall set of primitive edits (see Table 4), and $P(D) = 0.196$, is the probability of deny in the overall set (see Table 4).

If the overall set of primitive edits shown in Table 4 were divided by question A, we have the following.

The entropy of the primitive edits with answer to question A being 0 is

$$E(A=0) = -P(C/A=0) * \log_2^{P(C/A=0)} - P(D/A=0) * \log_2^{P(D/A=0)} = 0.37,$$

where $P(C/A=0) = 0.93$, is theconditional probability of clear in the primitive edits with answer to question A being 0, and $P(D/A=0) = 0.07$, is the conditional probability of deny in the primitive edits with answer to question A being 0.

Similarly, the entropy of the primitive edits with answer to question A being 1 $E(A=1) = 0.88$. And the entropy of the primitive edits partitioned by question A denoted as $E(A)$, is the weighted average of the two entropies given by

$$E(A) = P(A=0)E(A=0) + P(A=1)E(A=1) = 0.47,$$

where $P(A=0) = 0.8$, is the probability of answering NO to question A, and $P(A=1) = 0.2$, is the probability of answering YES to question A (see Table 3).

Similarly, we can calculate $E(B) = 0.60$ and $E(C) = 0.61$.

It is clear that the greatest entropy reduction can be realized if the overall set of primitive edits in Example 2 were partitioned according to answers to question A.

Step 2: Partition the overall set of primitive edits based on answers to the question A, which gives rise to the partial decision tree shown in Fig. 1.

Repeat steps 1 and 2 until all leaf nodes in the decision tree consist of primitive edits associated with a single outcome. We have the following decision tree as shown in Fig. 2.

For Heuristic 2, the expected number of questions to be asked is equal to the sum of the probabilities of each path multiplied by the number of arcs (each arc represents an answer to one of the questions) in that path. For example, the probability for path corresponding to "No" to question A and "No" to question C is given by $(0.9)\times(0.8) = 0.72$, as detailed in Table 3. The contribution of this path to the expected number of questions to be asked is $2\times0.72 = 1.44$. Continuing this procedure, the expected number of questions to be asked is found to be 2.08.

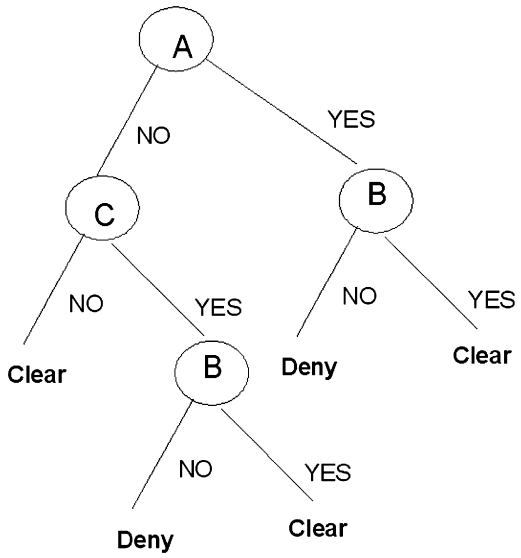

Fig. 2. Decision tree inducted from Example 2.

5. Performance analysis of heuristics

The heuristics to solving this problem was tested using randomly generated problem sets with some characteristics, which are reported in Tables 9 and 10. The results were intriguing and this section presents the details. It may be noted that both Heurisitcs 1 and 1a are static heuristics in the sense that the solution obtained gives a fixed sequence of questions to be asked. In this sense, the solution is easier to implement. However, solution produced by Heuristic 2 is dynamic, in the sense that the answers to the questions already posed will affect the questions to be asked. Implementation of this will require an online software to help the investigator. This may be borne in mind as we compare the performance of the heuristics.

There are three parameters in the problem, namely probability of answering "No" or "Yes" to each question (denoted as $P(\text{"No"}) = \varphi$, hence $P(\text{"Yes"}) = 1-\varphi$), the probability that the system will ultimately classify an entity as "Deny" or "Clear" (denoted as $P(\text{"Deny"}) = \chi$ and $P(\text{"Clear"}) = 1-\chi$) and the number of questions posed (denoted as n). In any large population, the number of suspicious entities may be assumed to be very small. Hence, the overall probability of "Deny" will be very small. Given a set of questions, the probability of getting an answer "Yes" or "No" from a population may depend on the question and the population. For example, if the question is: "Have you ever visited abroad?", the probability of "Yes" answer from New Yorkers may be higher compared to Iowans. For the sake of ease of comparison among problems and among solution procedures, we have set the probability of "Yes" or "No" to any question to be the same for all questions in a simulated problem set. This is clearly not a requirement for the solution procedures. In our experiments with the solution procedures, the parameters for $P(\text{"No"})$ comes from the set {0.01,0.1,0.5} and the P("Deny") is targeted (see explanation in the following paragraph) to come from the set {0.1, 0.01, 0.001}. The combination leads to 9 different problem sets. Problem sets were solved for cases with number of

Table 10
Performance of heuristics for P("No") = 0.01 and P("Deny") = 0.001.

Problem size (# of questions)	Expected number of questions to be asked based on five simulation runs			
	Heuristic 1	Heuristic 1a	Heuristic 2	Optimal solution by enumeration
6	3.414	4.064	4.042	3.059
9	4.658	6.118	5.454	4.315

Table 11
Performance of heuristics.

P("No")	P("Deny")	Problem size (number of questions) = 10		Problem size (number of questions) = 20	
		Heuristic 1a (expected number of questions to be asked based on five simulation runs)	Heuristic 2 (expected number of questions to be asked based on five simulation runs)	Heuristic 1a (expected number of questions to be asked based on five simulation runs)	Heuristic 2 (expected number of questions to be asked based on five simulation runs)
001	0.001	6.545	6.066	12.085	10.784
001	0.01	6.925	6.191	12.328	11.648
001	0.1	9.801	9.701	15.595	14.960
01	0.001	4.916	2.747	10.301	7.793
01	0.01	6.578	4.513	12.584	9.561
01	0.1	8.311	6.538	15.504	11.530
05	0.001	2.794	1.998	12.833	6.479
05	0.01	5.727	3.758	16.117	7.256
05	0.1	8.806	6.279	19.033	11.220

questions posed equal to 6 and 9 using all three heuristics and for problem sizes 10 and 20 using Heuristic 1a and Heuristic 2. It may be noted that Heuristic 1 may be used only for limited problem sizes due to computational burden. Next, we discuss the actual problem generation method used.

For the purpose of problem generation, the questions are numbered 1,2,…,n and are assumed to be posed in the natural order. Clearly there are 2^n sequences of answers. For each sequence, we calculate the probability of occurrence of that sequence based on the parameter φ. Suppose $n=3$, $\varphi=0.1$ and answer sequence is (1 1 0), then the probability of occurrence of answer sequence $(1\,1\,0)=(1-\varphi)\times(1-\varphi\ )\times\varphi=0.9\times0.9\times0.1=0.081$. For each answer sequence, we generate a classification based on the general distribution of "Deny" and "Clear". However, the overall probability of "Deny" in a given problem instance should be equal to the sum of the probabilities of occurrence of all answer sequences in the given instance that lead to the classification of "Deny", that is, it must be equal to χ. The random assignment procedure outlined earlier cannot guarantee that the overall probability of "Deny" will indeed equal to χ. Hence, we keep a continuous sum of the probability of "Deny" as we generate the problem instance and make the absolute value of the difference between the overall probability of "Deny" and χ as low as specified.

Our goal is to show that there is an enormous potential to save time and effort by appropriate sequencing of questions in an enquiry. The problem parameters were selected to reflect possible real life scenarios that may find applications for this formulation. For smaller problem sizes all three methods were used and performance analyzed on a comparative basis. However, Heuristic 1 would be too burdensome computationally for larger problems. The remaining two (Heuristics 1a and 2) are fairly robust and could be used for any conceivable real life scenario. For each problem set, 5 random instances were generated and all three solution procedures were applied to the instances and the results are reported in Table 10. As can be seen from the result, Heuristic 1 is the best performing heuristic and has cut down the expected number of questions to be asked by almost 50%. The performance of Heuristic 1 is also close to the performance of the optimal sequencing of questions, under which the expected number of questions to be asked is 3.059 for the problem size of 6. The optimal solution is found through exhaustive enumeration of all possible sequencing of questions. Execution time of each heuristic and the optimal solution were also recorded. The heuristics and the optimal solution were run on a PC with 1.83 GHz CPU and 1 G memory. For the problem of sequencing 6 questions, the average execution time of Heuristics 1, 1a, and 2, and the optimal solution is 2, 1, 1, and 32 s, respectively. For the problem of sequencing 9 questions, the average execution time of Heuristics 1, 1a, and 2, and the optimal solution is 303, 1, 1, and 3395 s, respectively. It may be noted that the order of magnitude of computational times for Heuristics 1a and 2 is about the same.

A more elaborate testing scheme was implemented for direct comparison of Heuristic 1a and 2 for larger problem sizes. The results are tabulated in Table 11. In general, classification based Heuristic 2 outperforms Heuristic 1a which is based on truncated enumeration. The savings in the expected number of questions to be asked using Heuristic 2 varies from 40% to 80%, with the exception of one combination (parameter set: P("No") = 0.01, and P("Deny") = 0.1, and problem size = 10). Using Heuristic 1a, the savings is typically 20–50% with some occasional exceptions. It may be noted that when the number of questions to be asked is relatively smaller and the P("Deny") is relatively larger, the two types of edits (clear and deny) are so wide-spread that it limits the ability of the algorithm to combine edits. It may be noted that if we keep P("No") constant, the performance of both heuristics deteriorates as the P("Deny") increases. As the probability of "Deny" increases, the answers become more mixed, reducing our ability to classify, which forces us to ask more questions before becoming certain about the outcome of the subject being questioned. Another approach to analyze the results of Table 11 would be to study the results by keeping the P("Deny") constant and comparing the results for changes in P("No"). It may be noted that the performance of both algorithms improve as the P("No") increases. The reason behind this trend is that for a given P("Deny"), fewer edits will be marked as "D" as the P("No") increases. This enables the algorithms to combine more primitive edits to higher order edits in Heuristic 1a. In the case of Heuristic 2, it represents lower level of diversity leading to improved results. However, the singular exception to this trend happens for Heuristic 1a for P("No") = 0.5. This is due to the fact that when the probability of occurrence of primitive

edits is equal, advantage of combining primitive edits with higher probability of occurrence first (Step 2 of Heuristic 1a) is lost, leading to poorer performance. Real life problems are unlikely to be of this type.

The data set of primitive edits is an imbalanced data set, where primitive edits with "Clear" decision greatly outnumbers primitive edits with "Deny" decision. One way of improving the performance of Heuristic 2 is to balance the data set by giving primitive edits with "Deny" decision more weight when calculating entropy. We experimented with this approach and solved problems of sizes 10–14 and weights in the range 25–200. In every instance, there was a slight improvement when compared to Heuristic 2, but not sufficient to warrant a separate report. There may be other approaches to deal with the inherent imbalance in the data set of the PSP, which is left as a future research question.

6. Conclusions and future research

A problem that is of great contemporary interest has been studied from the point of view of minimizing airport delays while maintaining the likelihood of detecting miscreants. Two areas of research readily come to mind, when we look for additional applications for this model and the heuristics. First is the problem of test design, where a decision to pass or fail a test-taker is based on the complete answer set. However, by ordering the questions optimally, one may be able to make the decision with fewer questions. The second application is in quality control, where we would like to minimize the number of tests performed, by ordering the tests appropriately to be able to make a final decision to accept or reject a supplied lot. The problem also appears to hold great promise for research into an area that clearly overlaps with the highly topical areas of data validation, editing, data integrity, and error checking [10,11].

In this paper, we have modeled airport security clearance problem as a question sequencing problem with an optimization criterion. We have developed three heuristics for the problem and report on their performance. An extension of interest would be the case where questions have three or more possible answers. Further, the answers may be in any measurement scale (nominal, ordinal, interval or ratio). As a pre-processing step, answers in interval or ratio scale will have to be broken into meaningful categories and converted to an ordinal scale. In any case, it would be difficult to apply the current version of Heuristics 1 and 1a to such a problem. One possible approach would be re-define general edits to account for three or more answers to each question. An alternative approach would be to recast the question into a series of binary questions. It is well-known that any question with t answers may be converted to an equivalent set of $\lceil \log_2 t \rceil$ binary questions. Such a conversion will allow us to apply Heuristics 1 and 1a to this problem. However, Heuristic 2 may be applied directly to such problems. It may also be noted that the static Heuristics 1 and 1a offer scope to modify them into a dynamic heuristic where the answers to the questions posed will modify the order of the future questions. These are posed as interesting future research questions. This will undoubtedly improve performance while complicating implementation. Another area of research is in the update of parameters. In this paper, it is assumed that the answers to the questions are independent. However, in real-life they may be dependent. That is, as the subject answers questions, one may be able to classify the subject more accurately as belonging to certain groups. Each of these groups may have different parameters for P("Yes") and P("No") to questions yet to be asked. Clearly these probabilities affect the expected number of questions to be asked. A dynamic update of these probabilities may lead to different questions to be asked in order to minimize the expected number of questions to be asked. This class of problems seems intrinsically hard, though we have not studied it from the point of view of problem complexity. One avenue of the future research would be to study whether they are NP-hard. Finally, all heuristics developed in this paper assume a given and static data set of questions and answers and their associated clear or deny. More advanced heuristics need to be developed for dynamic data sets with potential changing patterns of answers. Prior research on knowledge refreshing [6] could be employed for this line of future research. In summary, this is a very rich problem with a wide scope for algorithm development. This is also a real and contemporary problem which needs to be brought to the attention of researcher which may help spawn further research.

References

[1] Adam NR, Atluri V, Koslowski R, Grossman R, Janeja VP, Warner J. Secure interoperation for effective data mining in border control and homeland security applications. In: The seventh annual international conference on digital government research (dg.o 2006), San Diego, CA; May 21–24, 2006.

[2] Allison L. Information-theoretic sequence alignment. Technical Report 98/14, School of Computer Science and Software Engineering, Monash University, Australia 3168; June 1998.

[3] Chen H. Intelligence and security informatics: information systems perspective. Decision Support Systems 2006;41(3):555–9.

[5] Dewdney AK. The new turing omnibus: sixty-six excursions in computer science. W.H. Freeman and Company; 1993.

[6] Fang X, Rachamadugu R. Policies for knowledge refreshing in databases. Omega—The International Journal of Management Science 2009;37(1):16–28.

[7] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: Freeman; 1979.

[8] Gehrke J, Ramakrishnan R, Ganti V. RainForest: a framework for fast decision tree construction of large datasets. In: Proceedings of 1998 international conference of very large data bases (VLDB'98), New York, August 1998. p. 416–27.

[9] Hauck R, Atabakhsh H, Onguasith P, Gupta H, Chen H. Using coplink to analyze criminal-justice data. IEEE Computer 2002;35:30–7.

[10] Klein BD, Rossin DF. Data quality in neural network models: effect of error rate and magnitude of error on predictive accuracy. Omega—The International Journal of Management Science 1999;27(6):569–82.

[11] Klein BD. Detecting errors in data: classification of the impact of base rate expectations and incentives. Omega—The International Journal of Management Science 2001;29(5):391–404.

[12] Lin S, Brown DE. An outlier-based data association method for linking criminal incidents. Decision Support Systems 2006;41(3):604–15.

[13] Marshall B, Kaza S, Xu J, Atabakhsh H, Petersen T, Violette C, et al. Cross-jurisdictional criminal activity networks to support border and transportation security. In: seventh international IEEE conference on intelligent transportation systems, Washington, DC, 2004.

[14] Mehta M, Agrawal R, Rissanen J. SLIQ: a fast scalable classifier for data mining. In: Proceedings of 1996 international conference on extending database technology (EDBT'96), Avignon, France, March 1996.

[15] National Strategy for Homeland Security, Office of Homeland Security, The White House, ⟨http://www.whitehouse.gov/homeland/book/nat_strat_hls.pdf⟩, 2002.

[16] Quinlan JR. Introduction of decision tree. Machine Learning 1986;1:81–106.

[17] Quinlan JR. C4.5: programs for machine learning. San Mateo, CA: Morgan Kaufmann; 1993.

[18] Schaeffer GA, Briel JB, Fowles ME. Psychometric evaluation of the new GRE writing assessment. GRE Board Professional Report no. 96-11P; April 2001.

[19] Shafer J, Agrawal R, Mehta M. SPRINT: a scalable parallel classifier for data mining. In: Proceedings of 1996 international conference on very large data bases (VLDB'96), Bombay, India, September 1996. p. 544–55.

[20] Van Der Linden WJ. A maximin model for test design with practical constraints. Psychometrika 1989;54(2):237–47.

[21] Wainier H, Wang X. TOEFL Technical Report, TR-16, Educational Testing Service, Princeton, NJ; May 2001.

[22] Wang G, Chen H, Atabakhsh H. Automatically detecting deceptive criminal identities. Communications of the ACM 2004;47(3):71–6.

[23] Warner J, Atluri V, Mukkamala R. A credential-based approach for facilitating automatic resource sharing among ad-hoc dynamic coalitions. In: Proceedings of 19th annual IFIP WG 11.3 working conference on data and applications security, Storrs, CT, USA, August 7–10, 2005.

[24] Wu H, Gordon M, Demaagd K, Fan W. Mining web navigations for intelligence. Decision Support Systems 2006;41(3):574–91.