Published in:

Information Sciences, 94, pp. 87-118, 1996.

Interactive Learning of Monotone Boolean Functions

Last revision: March 24, 1996

BORIS KOVALERCHUK, EVANGELOS TRIANTAPHYLLOU[1], ANIRUDDHA S. DESHPANDE
Department of Industrial and Manufacturing Systems Engineering, 3128 CEBA Building, Louisiana State University, Baton Rouge, LA 70803-6409, U.S.A.
E-mail: ietrian@lsuvm.sncc.lsu.edu
Web: http://www.imse.lsu.edu/vangelis

and

EUGENE VITYAEV
Institute of Mathematics, Russian Academy of Science, Novosibirsk, 630090, Russia.

ABSTRACT
This paper presents some optimal interactive algorithms for some problems related to learning of monotone Boolean functions. These algorithms are based on the fundamental Hansel theorem [11]. The advantage of the algorithms is that they are not heuristics, as is often the case of many known algorithms for general Boolean functions, but they are optimal in the sense of the Shannon function. This paper also formulates a new problem for the joint restoration of two nested monotone Boolean functions f_1 and f_2. This formulation allows to further decrease the dialogue with an expert and restore non-monotone functions of the form $f_2 \& \neg f_1$. The effectiveness of the proposed approaches is demonstrated by some illustrative computational experiments.

1. INTRODUCTION

An important problem in machine learning is the one which involves the inference of a Boolean function from collections of positive and negative examples [3] [28]. This is also called the **logical analysis problem** [4] [10] and it is a special case of inductive inference. This kind of knowledge extraction is desirable when one is interested in deriving a set of rules which, in turn, can be easily comprehended by a field expert. In many application domains the end users do not have sophisticated computer and modeling expertise. As result, systems which are based on techniques such as neural networks, statistics, or linear programming, are not appealing to

[1] *The first two authors gratefully acknowledge the support from the* ***Office of Naval Research (ONR)*** *grant* ***N00014-95-1-0639****.*

them nor these methods provide a plausible explanation of the underlying decision-making process.

On the other hand, a logical analysis approach, when it is applicable, can result in rules which are already known to the end user (thus increasing his/her confidence on the method) or lead to new discoveries. In other words, a logical analysis approach has the distinct advantage of high comprehensibility when it is compared with other methods and to the discovery of new explicit knowledge. This is an important step on the way to scientific explanation of a studied phenomenon in comparison with a *"black box"* type of approach.

The most recent advances in distinguishing between elements of two pattern sets can be classified into six distinct categories. These are: a clause satisfiability approach to inductive inference by Kamath *et al.* [13] [14]; some branch-and-bound approaches of generating a small set of logical rules by Triantaphyllou *et al.* [24], and Triantaphyllou [25]; some improved polynomial time and NP-complete cases of Boolean function decomposition by Boros *et al.* [4]; linear programming approaches by Mangasarian *et al.* [22]; some knowledge based learning approaches by combining symbolic and connectionist (neural networks) machine based learning as proposed by Shavlik [23], Fu [7], Goldman *et al.* [8] and Cohn *et al.* [5] and finally, some nearest neighbor classification approaches by Hattoru and Torii [12], Kurita 21], Kamgar-Parsi and Kanal [15]. From the above six categories, the first three can be considered as logical analysis approaches, since they deal with inference of Boolean functions.

The main challenge in inferring a target Boolean function from positive and negative examples is that *the user can never be absolutely certain about the correctness of the inferred function, unless he/she has used the entire set of all possible examples* which is of size 2^n. Apparently, even for a small value of n, this task may be practically impossible to realize [1]. Fortunately, many real life applications are governed by the behavior of a **monotone** system or can be described by **a combination of a small number of monotone systems**. In machine learning monotonicity offers some unique computational advantages [19]. By knowing the class of certain examples, one can easily infer the class membership of more examples. This, in turn, can significantly expedite the learning process.

2. SOME BASIC DEFINITIONS AND RESULTS ABOUT MONOTONE BOOLEAN FUNCTIONS

Let E_n denote the set of all binary vectors of length n. Let x and y be two such vectors. Then, the vector $x = (x_1,x_2,x_3,\ldots,x_n)$ **precedes** the vector $y = (y_1,y_2,y_3,\ldots,y_n)$ (denoted as: $x \preceq y$) if and only if the following is true: $x_i \leq y_i$, for all $1 \leq i \leq n$.

A Boolean function $f(x)$ is **monotone** if for any vectors $x,y \in E_n$, the relation $f(x) \le f(y)$ follows from the fact that $x \preceq y$. Let M_n be the set of all monotone Boolean functions defined on n variables. A binary vector x of length n is said to be an **upper zero** of a function $f(x) \in M_n$, if $f(x) = 0$ and, for any vector y such that $x \prec y$, we have $f(y) = 1$. Also, we shall call the number of unities (i.e., the number of the *"1"* elements) in vector x as its **level** and denote this by $U(x)$. An upper zero x of a function f is said to be the **maximal upper zero** if we have $U(y) \le U(x)$ for any upper zero y of the function f (e.g., Kovalerchuk and Lavkov [18]). In an analogous manner, we can define the concepts of **lower unit** and **minimal lower unit**. A binary vector x of length n is said to be a **lower unit** of a function $f(x) \in M_n$, if $f(x) = 1$ and, for any vector y from E_n such that $y \prec x$, we get $f(y) = 1$. A lower unit of a function f is said to be the **minimal lower unit** if we have $U(x) \le U(y)$ for any lower unit y of the function f.

For the **number** $\psi(n)$ of **monotone Boolean functions** depending on n variables, it is known that:

$$\psi(n) = 2^{\binom{n}{\lfloor n/2 \rfloor}(1+\varepsilon(n))}$$

where $0 < \varepsilon(n) < c(\log n)/n$ and c is a constant (see, for instance, [2] and [16])). Let a monotone Boolean function f be defined with the help of a certain operator A_f (also called an **oracle**) which when fed with a vector $x = (x_1,x_2,x_3,\ldots,x_n)$, returns the value of $f(x)$. Let $\mathcal{F} = \{F\}$ be the set of all algorithms which can solve the above problem and $\varphi(F, f)$ be the number of accesses to the operator A_f required to solve a given problem about inferring a monotone function $f \in M_n$.

Next, we introduce the Shannon function $\varphi(n)$ as follows [17]:

$$\varphi(n) = \min_{F \in \mathcal{F}} \max_{f \in M_n} \varphi(F, f). \qquad (2\text{-}1)$$

The problem examined next is that of finding all maximal upper zeros (lower units) of an arbitrary function $f \in M_n$ with the help of a certain number of accesses to the operator A_f. It is shown in [11] that in the case of this problem the following relation is true (known as ***Hansel's theorem***):

$$\varphi(n) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}. \qquad (2\text{-}2)$$

Here $\lfloor n/2 \rfloor$ is the closest integer number to $n/2$ which is no greater than $n/2$. In terms of machine learning the set of all maximal upper zeros represents the **border elements** of the negative pattern. In an analogous manner the set of all minimal lower units to represent the border of a positive pattern. In this way a monotone Boolean function represents two *"compact patterns"*. Restoration algorithms for monotone Boolean functions which use Hansel's theorem are optimal in terms of the Shannon function. That is, they minimize the maximum time

requirements of any possible restoration algorithm. It is interesting to note at this point that, to the best of our knowledge, Hansel's theorem has not been translated into English, although there are numerous references of it in the non English literature (Hansel wrote his paper in French). This theorem is a one of the final results of the long term efforts in monotone Boolean functions beginning from Dedekind in 1897 [6] (in German).

One approach to use monotonicity in combination with existing pattern recognition/classification approaches is to consider each one of the available examples and apply the concept of monotonicity to generate many more data. However, such an approach would have to **explicitly** consider a potentially humongous set of input observations. On the other hand, these classification approaches are mostly NP-complete and very CPU time consuming. Thus, such an approach would be inefficient.

3. SOME KEY PROBLEMS AND ALGORITHMS

Problem Description: *The nested classification problem*

When we have one classification problem we should learn to classify an arbitrary vector as a result of the interaction of the designer with an *"oracle"* (i.e., an operator A_f). Now we assume that we have two classification problems, i.e., we need to restore two Boolean functions f_1 and f_2, to discriminate classes. We want to solve these classification problems simultaneously, by interacting with two oracles A_{f1} and A_{f2}. We also assume that these two problems are **nested**, i.e. all positive cases for f_1 are also positive cases for f_2. That is, for all $\alpha \epsilon E_n$, the following relation is always true: $f_2(\alpha) \geq f_1(\alpha)$. We also accompany this assumption with the requirement of monotonicity of f_1 and f_2.

An Illustrative Example of Nested Problems: Breast Cancer Diagnosis

For this illustrative example consider the following nested classification problem related to breast cancer diagnosis. The **first sub-problem** is related to the clinical question of whether a biopsy or short term follow-up is necessary or not. The **second sub-problem** is related to the question whether the radiologist believes that the current case is highly suspicious for malignancy or not. It is assumed that if the radiologist believes that the case is malignant, then he/she will also definitely recommend a biopsy. More formally, these two sub-problems are defined as follows:

The Clinical Management Sub-Problem: One and only one of the following two disjoint outcomes is

possible:

1) *"Biopsy/short term follow-up is necessary"*, or:

2) *"Biopsy/short term follow-up is not necessary"*.

The Diagnosis Sub-Problem: Similarly as above, one and only one of two following two disjoint outcomes is possible. That is, a given case is:

1) *"Highly suspicious for malignancy"*, or:

2) *"Not highly suspicious for malignancy"*.

Our goal here is to extract the way the system operates in the form of two discriminant Boolean functions f_2 and f_1. The first function is related to the first sub-problem, while the second function is related to the second sub-problem. Also observe, that the following relations must be true: $E^+_1 \supset E^+_2$ and $f_1(\alpha) \geq f_2(\alpha)$ for all $\alpha \in E_n$, describing the system state, where $f_1(\alpha)$ and $f_2(\alpha)$ are the discriminant monotone Boolean functions for the first and second task, respectively. Next the outline of an algorithm for solving this kind of problems is presented.

Algorithm A1:

Step 1: The user is asked to confirm the monotonicity for both tasks (i.e., the monotonicity of the functions underlying the patterns related to the previous two tasks or sub-problems). The user is also required to confirm that the two functions are nested (i.e., $E^+_1 \supseteq E^+_2$ and $f_1(\alpha) \geq f_2(\alpha)$ for all $\alpha \in E_n$).

Step 2: If the user confirmed that monotonicity should be present, then test for the monotonicity property in the initial examples for both tasks. Also, confirm that the available examples satisfy the *"nesting"* requirement.
If the user did not confirmed monotonicity or the *"nesting"* requirement, then he must reformulate the problem characteristics (features) to reach monotonicity.
If monotonicity is not reached, then do not solve the problem with monotone Boolean functions and exit.

Step 3: Reject the examples which violate monotonicity. At this point the following sets of examples are available: E^+_1, E^-_1 and E^+_2, E^-_2, and $E^+_1 \supseteq E^+_2$ should hold.

Step 4: Use the algorithm in [9] (which is briefly described in section 4.2.) to create all Hansel chains for the number (denoted as n) of features in the current application.

Step 5: Use the initial (i.e., already classified) examples and the monotonicity property to infer all derivable classification values on the additional examples located in the above Hansel chains. For effective storage of the inferred examples use the approach described in [18]. These ideas use the property of Hansel chains according to which at each iteration i we need to store only two key examples from each Hansel chain. These two examples represent a maximal known upper zero and a minimal known lower unit for each Hansel chain.

Step 6: Generate two new candidate examples, say $\alpha^{(1)}_i$ and $\alpha^{(2)}_i$ to be considered for classification by oracles A_{f1} and A_{f2}, respectively. These examples are taken from Hansel chains according to the procedures described in section 4.2. If no more examples are left, exit; the two functions have been fully restored. Otherwise, go to step 7.

Step 7: Given the above two examples $\alpha^{(1)}_i$ and $\alpha^{(2)}_i$, we send for classification the one which leads to the maximum number of additional examples to be classified (without new calls to the oracles) when the properties of monotonicity and nested functions are applied. How this is done is described in more detail next.

Step 8: After either $\alpha^{(1)}_i$ and $\alpha^{(2)}_i$ has been classified by the appropriate oracle, use the monotonicity property to infer all derivable classification values of the additional examples located in the remaining Hansel chains. Go to step 6.

Next, we discuss step 7 in more detail. When step 7 is reached, there are two candidate examples, denoted as $\alpha^{(1)}_i$ and $\alpha^{(2)}_i$, which are considered for submission to the respective oracle. For the simultaneous restoration of f_1 and f_2 we need to send first for classification the example ($\alpha^{(1)}_i$ or $\alpha^{(2)}_i$) which best accelerates the learning process. Apparently, there are two possible alternatives: (i) first to ask oracle A_{f1} about the value of $f_1(\alpha^{(1)}_i)$ and (ii) first to ask the second oracle A_{f2} about the value of $f_2(\alpha^{(2)}_i)$. These alternatives allow to guarantee for both of oracles and functions f_1 and f_2 the optimum limits on the number of calls in (2-2). We select which example to submit first by computing two estimates N_1 and N_2 of the numbers of examples that can be classified if example $\alpha^{(1)}_i$ or $\alpha^{(2)}_i$ is classified first, respectively. Given the previous estimates, we decide which example to submit first for classification according to the following simple rules:

Rule 1: If $N_1 > N_2$, then ask A_{f1} about the $f_1(\alpha^{(1)}_i)$ value;
Rule 2: If $N_1 < N_2$, then ask A_{f2} about the $f_2(\alpha^{(2)}_i)$ value;
and **Rule 3:** If $N_1 = N_2$, then choose $\alpha^{(1)}_i$ or $\alpha^{(2)}_i$ randomly.

In step 7 we realized a myopic algorithm, which can be extended. Recall that we considered the possibilities of classification values $f_1(\alpha^{(1)}_i)$ and $f_2(\alpha^{(2)}_i)$ at the next level (denoted as the i-th level). One may wish to extend this to the $(i+1)$ level and so on. In this way the estimates will be more accurate (this is similar to playing chess by considering two, three or more moves in advance). However, this may significantly increase the time requirements.

4. AN ALGORITHM FOR RESTORING A MONOTONE BOOLEAN FUNCTION

4.1. General scheme of the algorithm.

Next we present algorithm **RESTORE** for the interactive restoration of a monotone Boolean function and two procedures **GENERATE** and **EXPAND** in a pseudo programming language.

ALGORITHM "RESTORE" $f(x_1,x_2,\ldots,x_n)$

Input. Dimension n of the binary space and access to an oracle A_f (which has to be inferred).

Output. A monotone Boolean function restored after a minimal number (according to Shannon criterion as it is given as relation (2-2) in section 2) of calls to the oracle A_f.

Method.

1) Construction of Hansel chains (see section 4.2)

2) Restoration of a monotone Boolean function starting from chains of minimal length and finishing with chains of maximal length. This ensures that the number of membership calls to the oracle A_f is no more than the limit presented in formula (2-2), which guarantees the number of calls to an oracle A_f to be no more than the limit presented in formula (2-1).

```
BEGIN ;
  Set i=1; {initialization}
  DO WHILE (function f(x) is not entirely restored)
     Step 1:  Use procedure GENERATE to generate element α_i;
     Step 2:  Call oracle A_f to retrieve the value of f(α_i);
     Step 3:  Use procedure EXPAND to deduce the values of other elements in Hansel chains (i.e., sequences
              of examples in E_n) by using the value of f(α_i ), the structure of element α_i and the monotony
              property.
     Step 3:  Set  i ← i+1;
  RETURN
END;
```

PROCEDURE "GENERATE": Generate i-th element α_i to be classified by the oracle A_f.

Input. Dimension n of the binary space.
Output. The next element to send for classification by the oracle A_f.
Method: Proceeding from minimal to maximal Hansel chains.

```
IF i=1  THEN  {where i is the index of the current element}
  BEGIN
     Step 1.1:  Retrieve all Hansel chains of minimal length;
     Step 1.2:  Randomly choose the first chain C_1 among the chains retrieved in step 1.1;
     Step 1.3:  Set the first element α_1  as the minimal element of chain C_1;
  END
```

ELSE
BEGIN
Set $k=1$ {where k is the index number of a Hansel chain};
DO WHILE (NOT all Hansel chains are tested)

Step 2.1: Find the largest element α_i of chain C_k, which still has no confirmed $f(\alpha_i)$ value;

Step 2.2: If step 2.1 did not returned an element α_i then randomly select the next Hansel chain C_{k+1} of the same length as the one of the current chain C_k;

Step 2.3: Find the least element α_i from chain C_{k+1}, which still has no confirmed $f(\alpha_i)$ value;

Step 2.4: If Step 2.3 did not return an element α_i, then randomly choose chain C_{k+1} of the next available length;

Step 2.5: Set $k \leftarrow k+1$;

RETURN
END

PROCEDURE "EXPAND": Assign values of $f(x)$ for $x \geq \alpha_i$ or $x \leq \alpha_i$ in chains of a given length l and in the chains of the next length $l+2$.

Input. The $f(a_i)$ value.

Output. An extended set of elements with known $f(x)$ values.

Method: The method is based on the properties: if $x \geq \alpha_i$ and $f(\alpha_i)=1$, then $f(x)=1$; and if $x \leq \alpha_i$ and $f(\alpha_i)=0$, then $f(x)=0$.

BEGIN

Step 1: Obtain x such that $x \geq \alpha_i$ or $x \leq \alpha_i$ and x is in a chain of the length l or $l+2$.

Step 2: If $f(\alpha_i)=1$, then $\forall x\ (x \geq \alpha_i)$ set $f(x)=1$;
If $f(\alpha_i)=0$, then $\forall x\ (x \leq \alpha_i)$ set $f(x)=0$;

Step 3: Store the $f(x)$ values which were obtained in step 2;

END

Comment: Effectiveness of these procedures is based on the proof of Hansel lemma [11].

4.2. Construction of Hansel Chains

Several steps in the previous algorithms deal with Hansel chains. Next we describe how to construct all Hansel chains for a given state space E_n of dimension n. First, we give a formal definition of a general chain.

A **chain** is a sequence of binary vectors (examples) $\alpha_1, \alpha_2, \ldots \alpha_i, \alpha_{i+1}, \ldots, \alpha_q$, such that α_{i+1} is obtained from a_i by changing a "0" element to "1". That is, there is an index k such that $\alpha_{i,k} = 0$, $\alpha_{i+1,k} = 1$ and for any $t \neq k$ the following is true $\alpha_{i,t} = \alpha_{i+1,t}$. For instance, the following list $<01000, 01100, 01110>$ of three vectors is a chain. To construct all Hansel chains we will use an iterative procedure as follows: Let $E_n = \{0,1\}^n$ be the n-dimensional binary cube. All chains for E_n are constructed from chains for E_{n-1}. Therefore, we begin the construction for E_n by starting with E_1 and iteratively proceeding to E_n.

(i) **Chains for E_1.**

For E_1 there is only a single (trivial) chain and it is $<(0), (1)>$. Note that (0) is the first element (vector of size 1) while (1) is the second element.

(ii) **Chains for E_2.**

First we consider E_1 and add at the beginning of each one of its chains the element (0). Thus, we obtain the set $\{00, 01\}$. This set is called E_2^{min}. Similarly, by adding "1" to $E_1 = \{(0),(1)\}$ we construct the set $E_2^{max} = \{10, 11\}$. To simplify notation often we will omit "( )" for vectors as (10) and (11). Both E_2^{min} and E_2^{max} are isomorphic to E_1, because they have the isomorphic chains $<00, 01>$ and $<10, 11>$. The union of E_2^{min} and E_2^{max} is E_2. That is, $E_2 = E_2^{min} \cup E_2^{max}$. Observe that the chains $<00, 01>$ and $<10, 11>$ cover entirely E_2, however, they are not Hansel chains. To obtain Hansel chains we need to modify them as follows.

Adjust the chain $<00, 01>$ by adding the maximum element (11) from the chain $<10, 11>$. That is, obtain a new chain $<00, 01, 11>$. Also reject element (11) from the chain $<10, 11>$. Hence, we obtain the two new chains: $<00, 01, 11>$ and $<10>$. These chains are the **Hansel chains** [11] for E_2. That is $E_2 = \{<00, 01, 11>, <10>\}$.

(iii) **Chains for E_3.**

The Hansel chains for E_3 are constructed in a manner similar to the chains for E_2. First, we double and adjust the Hansel chains of E_2 to obtain E_3^{min} and E_3^{max}. The following relations is also true:

$$E_3 = E_3^{min} \cup E_3^{max},$$

where $E_3^{min} = \{<000, 001, 011>, <010>\}$ and $E_3^{max} = \{<100, 101, 111>, <110>\}$. We do the same chain modification as for E_2. That is, first we choose two isomorphic chains. At first, let it be two maximal length chains $<000, 001, 011>$ and $<100, 101, 111>$. We add the maximal element (111) from $<100, 101, 111>$ to $<000, 001, 011>$ and drop it from $<100, 101, 111>$. In this way we obtain the two new chains: $<000, 001, 011, 111>$ and $<100, 101>$. Next, we repeat this procedure for the rest of the isomorphic chains $<010>$ and $<110>$. In this simple case we will have just one new chain $<010, 110>$ (note that the second chain will be empty). Therefore, E_3 consists of the three Hansel chains $<010, 110>$, $<100, 101>$ and $<000, 001, 011, 111>$. That is, the following is true:

$$E_3 = \{<010, 110>, <100, 101>, <000, 001, 011, 111>\}.$$

In the similar manner one can construct the Hansel chains for E_4, E_5 and so on. Note that the Hansel chains of E_{n-1} can be obtained recursively from the Hansel chains of E_1, E_2, …, E_{n-2}. Figures 1 and 2 depict the above issues for the E_2, E_2 and E_3 spaces.

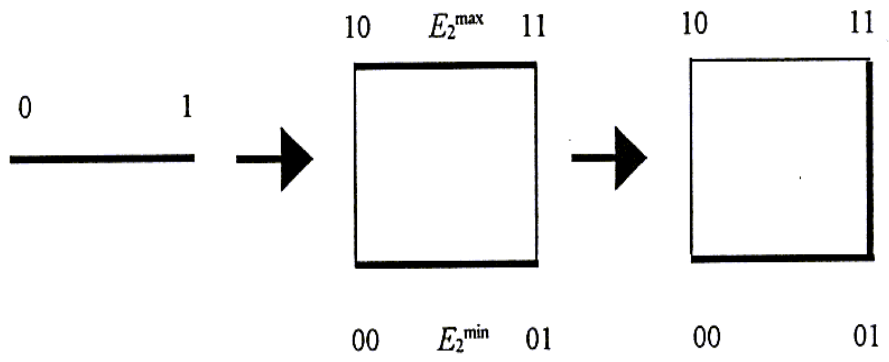

Fig 1. Construction of Hansel chains for E_2.

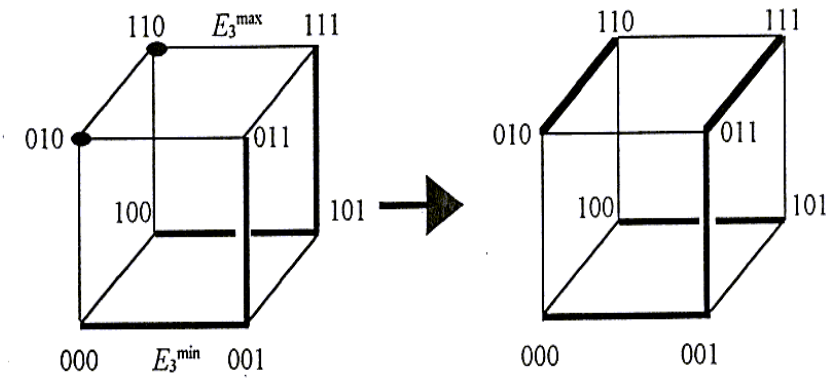

Fig 2. Construction of Hansel chains for E_3.

5. A COMPUTATIONAL INTERACTIVE HIERARCHICAL EXPERIMENT

5.1 Problem Description

In this section we demonstrate the previous issues in terms of an illustrative example from the area of breast cancer diagnosis. It is assumed that some preliminary medical tests have been performed (such as a mammogram) and the medical doctor wishes to determine what to do next. That is, should he/she proceed with a biopsy or not? The problem considered in this section is a **nested** one because it is comprised by two

interrelated sub-problems. The first problem is related to the question whether a biopsy is necessary. The second sub-problem is related to the question whether the human expert (medical doctor) believes that the current case represents a malignant case or not. It is assumed that if the doctor believes that the case is malignant, then he/she will also definitely recommend a biopsy.

The previous two interrelated sub-problems can be formulated as the restoration problem of two nested monotone Boolean functions. A medical expert was explained the concept of monotonicity and he felt comfortable with the idea of using monotone Boolean functions. Moreover, the dialogue which followed confirmed the validity of this assumption. The underlying Boolean functions were defined on 11 binary atoms. This simplification was done to illustrate the applicability of the proposed methodology. More details on this problem can be found in [20].

From the above discussion it follows that now we will work in the 11-dimensional binary space (i.e., in E_{11}). The two corresponding functions are f_1 and f_2. Function f_1 returns true (1) value if the decision is *"biopsy/short term follow-up is necessary"*, false (0) otherwise. Similarly, function f_2 returns true (1) value if the decision is *"highly suspicious for malignancy"*, false (0) otherwise.

Full restoration of either one of the functions f_i (for $i = 1,2$) without any optimization on the dialogue process would had required up to $\mathbf{2^{11}=2,048}$ calls (membership inquires) to an expert (i.e., the A_f operator or medically oriented binary attributes. However, according to the Hansel lemma (i.e., relation (2-2) in section 2) and under the assumption of monotony, an optimal (i.e., with a minimal number of accesses) dialogue for restoring a monotone Boolean would require at most: $\binom{11}{5} + \binom{11}{6} = 2 \times 462 = 924,$ calls to the oracle. Please observe that this new value is by 2.36 times smaller than the previous upper limit of 2,048 calls. However, even this upper limit of 924 calls can be reduced even further, as it explained in the next section.

5.2. Hierarchical Decomposition

In this particular breast cancer diagnosis problem the medical expert indicated that the original 11 binary attributes can be organized in terms of a **hierarchy**. This hierarchy follows from the definition of the 11 medically oriented binary attributes. Next, it will be shown that this hierarchy can be exploited to reduce the required number of calls to the *"oracle"* even further. We will not describe the medical justification of this hierarchy here (more details can be found in [20]). The hierarchy used in this example is given as follows:

Level 1 (5 attributes) **Level 2 (all 11 attributes)**

x_1	←	$w_1,\ w_2,\ w_3$
x_2	←	$y_1,\ y_2,\ y_3,\ y_4,\ y_5$
x_3	←	x_3
x_4	←	x_4
x_5	←	x_5

In this hierarchy the two attributes x_1 and x_2 correspond to two monotone Boolean functions which also have to be restored. That is, the following is true:

$$x_1 = \varphi(w_1,\ w_2,\ w_3),$$

and $$x_2 = \psi(y_1,\ y_2,\ y_3,\ y_4,\ y_5).$$

The expert indicated that these two functions φ and ψ should be common to both problems (i.e., the biopsy and cancer problems). Therefore, the following relation is true regarding the f_i (for $i = 1, 2$) and the two φ, and ψ functions:

$$f_i(x_1,x_2,x_3,x_4,x_5) = f_i(\varphi(w_1,w_2,w_3),\ \psi(y_1,y_2,y_3,y_4,y_5),\ x_3,x_4,x_5)).$$

Given the above analysis and decomposition, then according to Hansel theorem (i.e., relation (2-2) in section 2) it follows that for the restoration problem at level 1 it would had been required to have at most $\binom{5}{2} + \binom{5}{2} = 20$, calls to the *"oracle"*. Observe that without the exploitation of the monotonicity property, an exhaustive dialogue requires $2^5=32$ calls which is **by 1.6 times higher.** Also the exploitation of a hierarchy can even benefit the restoration of general (i.e., not necessarily monotone) Boolean functions.

The above bounds represent upper limits. When we actually interviewed the medical expert, the target function f_1 was fully restored with only 13 membership inquires (calls). Therefore, the number of the actual calls was 1.5 times less than the upper limit given by Hansel's theorem and 2.5 times less than the exhaustive search (which would had required up to 32 calls). Also, the same 13 calls were adequate to restore $f_2(x_1,x_2,x_3,x_4,x_5)$ and 12 calls were adequate to restore $x_2 = \psi(y_1,y_2,y_3,y_4,y_5)$.

Next x_1 was restored as a function $\varphi(w_1,w_2,w_3)$. In this case the exhaustive search would had required $2^3 = 8$ calls. On the other hand, Hansel's lemma provides an upper limit of $\binom{1}{3} + \binom{2}{3} = 6$, calls. This value is by 1.33 times less than what the exhaustive search would had required. Although the previous numbers of calls are not excessively high, they still provide a sense of the potential of significant savings which can be achieved if monotonicity is established and utilized. The numbers in this example are supposed only to be for illustrative purposes in clarifying the proposed methodology.

In the following paragraphs we present the optimal dialogue in restoring the $\varphi(w_1,w_2,w_3)$ function. Recall

from section 4.2 that the Hansel chains for the three dimensional space E_3 are $\{<010, 111>, <100, 101>, <000, 001, 011, 111>\}$. The medical expert was asked to evaluate the value of some of the above examples. Given the value of certain examples in these chains and using the monotonicity property, we were able to induce the values of more examples. The results are summarized below:

	Chain 1	**Chain 2**	**Chain 3**
Vectors:	<010, 110>	<100, 101>	<000, 001, 011, 111>
Value of φ	1* 1	0* 1*	0 0* 1 1

In the above illustration "1*" and "0*" indicate answers given directly by the expert. For instance, during the dialogue the expert decided that: φ(010) = 1, φ(101) = 1, and φ(100) = φ(001) = 0. The values without an asterisk (i.e., when we only have "1" or "0") were obtained by utilizing the monotonicity property.

In this experiment the four calls about the function values for the examples (010), (100), (101) and (001) were sufficient to lead to full restoration. Monotonicity allowed us to extend the previous four values to infer the values of the rest of the examples. For instance, from the expert's testimony that φ(010) = 1, it is derived that φ(110) = 1, and from φ(001)=0, it is derived that φ(000) = 0. Observe that these 4 calls to the expert are 2 times less than the maximum number of calls (i.e., 8) and 1.5 times less than the guaranteed number of calls provided by Hansel's theorem (which is equal to 6).

In order to describe function $\varphi(w_1, w_2, w_3)$ in the standard Disjunctive Normal Form (DNF) we should include from each one of the chains 1-3 above their minimal upper units, i.e. (010), (101) and (011) presented as w_2, w_1w_3, w_2w_3 respectively. As result we obtained their DNF expression. This DNF, after simplification, is: $w_2 \vee w_1w_3$. That is, the following is true:

$$x_1 = \varphi(w_1, w_2, w_3) = w_2 \vee w_1w_3 \vee w_2w_3 = w_2 \vee w_1w_3. \quad (5\text{-}1)$$

In a similar manner we can obtained Boolean expressions for $x_2 = \psi(y_1, y_2, y_3, y_4, y_5)$ from the Hansel chains for E_5:

$$x_2 = \psi(y_1, y_2, y_3, y_4, y_5) = y_2 \vee y_1 \vee y_3y_4y_5. \quad (5\text{-}2)$$

Similarly for level 1 we obtained the target functions of x_1, x_2, x_3, x_4, x_5 for the **biopsy** sub-problem to be defined as follows:

$$f_1(x) = x_2x_4 \vee x_1x_2 \vee x_1x_4 \vee x_3 \vee x_5. \quad (5\text{-}3)$$

and for the **cancer** sub-problem to be defined as:

$$f_2(x) = x_1x_2 \vee x_3 \vee x_2x_5 \vee x_1x_5 \vee x_4x_5 = x_1x_2 \vee x_3 \vee (x_2 \vee x_1 \vee x_4)x_5. \quad (5\text{-}4)$$

The above expression allows to compare the two functions f_1 and f_2. Observe that $f_1(x) = A \vee (x_2 \vee x_1)x_4 \vee x_5$, and $f_2(x) = A \vee (x_2 \vee x_1 \vee x_4)x_5$, where $A = x_1x_2 \vee x_3$. Hence, these two functions differ only in the parts $(x_2 \vee x_1)x_4 \vee x_5$ and $(x_2 \vee x_1 \vee x_4)x_5$ which may be an important information for additional medical analysis. Next we use expressions (5-1) and (5-2) in (5-3) and (5-4) to define the explicit form f_1 and f_2 as the proposed final formulas for performing **biopsy**:

$$f_1(x) = x_2x_4 \vee x_1x_2 \vee x_1x_4 \vee x_3 \vee x_5 =$$

$$= (y_2 \vee y_1 \vee y_3y_4y_5)x_4 \vee (w_2 \vee w_1w_3)(y_2 \vee y_1 \vee y_3y_4y_5) \vee (w_2 \vee w_1w_3)x_4 \vee x_3 \vee x_5, \qquad (5\text{-}5)$$

and for **cancer**:

$$f_2(x) = x_1x_2 \vee x_3 \vee x_2x_5 \vee x_1x_5 \vee x_4x_5 =$$

$$= (w_2 \vee w_1w_3)(y_2 \vee y_1 \vee y_3y_4y_5) \vee x_3 \vee (y_2 \vee y_1 \vee y_3y_4y_5)x_5 \vee (w_2 \vee w_1w_3)x_5 \vee x_4x_5. \qquad (5\text{-}6)$$

Totally we needed <u>29 calls</u> to restore f_1 in the following form:

$$f_i(x_1,x_2,x_3,x_4,x_5) = f_i(\varphi(w_1,w_2,w_3), \psi(y_1,y_2,y_3,y_4,y_5),x_2,x_3,x_4,x_5)).$$

The same 29 calls were also needed to restore f_2 independently. In reality, however, we used 42 calls instead of 58 calls to restore <u>both functions</u> due to the fact that the component functions φ and ψ are the same for the target functions of both problems. For the record, we spent for the dialogue with the medical expert no more than **one hour**.

At this point it is interesting to observe that if one wishes to restore the **<u>non</u>** monotone Boolean function which corresponds to the concept: *"biopsy and not cancer"*, then this can be achieved easily as follows. First note that the above concept represents cases in which surgery can potentially be avoided. This concept can be presented with the composite formula: $f_1 \& \neg f_2$, and thus it can be computed by using expressions (5-5) and (5-6). Therefore, a total number of 42 calls is just sufficient to restore this function (which is both non monotone and very complex mathematically).

TABLE 1

Dialogue Sequences for Numerical Experiment

#	Vector	f_1	f_2	ψ	Extension		$f_1^{\sim}$	$f_2^{\sim}$	$f_1^{\wedge}$	$f_2^{\wedge}$
					1 → 1	0 → 0				
1	2	3	4	5	6	7	8	9	10	11
Chain 1.										
1.1	(01100)	1*	1*	1*	1.2;6.3;7.3	7.1;8.1	1*	1*	1	1*
1.2	(11100)	1	1	1	6.4;7.4	5.1;3.1	1	1	1	1
Chain 2.										
2.1	(01010)	1*	0*	1*	2.2;6.3;8.3	6.1;8.1	1*	0*	1*	0*
2.2	(11010)	1	1*	1	6.4;8.4	3.1;6.1	1	1*	1	1*
Chain 3.										
3.1	(11000)	1*	1*	1*	3.2	8.1;9.1	1	1*	1	1*
3.2	(11001)	1	1	1	7.4;8.4	8.2;9.2	1	1	1	1
Chain 4.										
4.1	(10010)	1*	0*	1*	4.2;9.3	6.1;9.1	1*	0*	1*	0*
4.2	(10110)	1	1*	1	6.4;9.4	6.2;5.1	1	1*	1	1*
Chain 5.										
5.1	(10100)	1*	1*	1*	5.2	7.1;9.1	1	1*	1	1*
5.2	(10101)	1	1	1	7.4;9.4	7.2;9.2	1	1	1	1
Chain 6.										
6.1	(00010)	0*	0	0*	6.2;10.3	10.1	0*	0	0*	0*
6.2	(00110)	1*	1*	0*	6.3;10.4	7.1	1	1*	1*	1*
6.3	(01110)	1	1	1	6.4;10.5		1	1	1	1
6.4	(11110)	1	1	1	10.6		1	1	1	1
Chain 7.										
7.1	(00100)	1*	1*	0*	7.2;10.4	10.1	1	1*	1	1*
7.2	(00101)	1	1	0*	7.3;10.4	10.2	1	1	1	1
7.3	(01101)	1	1	1*	7.4;10.5	8.2;10.2	1	1	1	1
7.4	(11101)	1	1	1	5.6		1	1	1	1
Chain 8.										
8.1	(01000)	0*	0	1*	8.2	10.1	0*	0	0*	0
8.2	(01001)	1*	1*	1	8.3	10.2	1	1*	1	1*
8.3	(01011)	1	1	1	8.4	10.3	1	1	1	1
8.4	(11011)	1	1	1	10.6	9.3	1	1	1	1
Chain 9.										
9.1	(10000)	0*	0	1*	9.2	10.1	0*	0	0*	0*
9.2	(10001)	1*	1*	1	9.3	10.2	1	1*	1*	1*
9.3	(10011)	1	1	1	9.4	10.3	1	1	1	1
9.4	(10111)	1	1	1	10.6	10.4	1	1	1	1
Chain 10.										
10.1	(00000)	0	0	0	10.2		0	0	0	0
10.2	(00001)	1*	0*	0	10.3		1*	0*	1*	0
10.3	(00011)	1	1*	0	10.4		1	1*	1	1*
10.4	(00111)	1	1	1	10.5		1	1	1	1
10.5	(01111)	1	1	1	10.6		1	1	1	1
10.6	(11111)	1	1	1			1	1	1	1
Total Calls		**13**	**13**	**12**			**7**	**13**	**22**	

5.3. Independent Search of Functions

Below we systematically present how we obtained the results in section 5.2. The main technical tool is table 1. This table represents the dialogue which was executed based on the algorithm for problem 1 (in section 3). The information in table 1 resulted to the formation of formulas (5-2) to (5-6). The first column of table 1 designates binary vectors (examples): the first number indicates a Hansel chain and the second number indicates the location of the vector within that chain. The second column presents binary vectors. The third, forth and fifth columns present values of the functions f_1, f_2 and ψ, respectively. Asterisks "*" show values obtained directly from the expert. Values without asterisks were inferred by using the previous values and the property of monotonicity.

The sixth column shows the indexes of vectors for which we can use monotony if a given function evaluates to 1 (i.e., true value). For instance, for vector #7.1 we have $f_1(00100) = 1^*$ (as given by the expert), hence we can also set (by utilizing the monotonicity property) $f_1(00101) = 1$, because vector (00101) is vector #7.2. We can also do the same with vector #10.4.

Similarly, the seventh column represents the indexes of the vectors for which we can use monotony if a given function evaluates to 0 (i.e., false value). For instance, for the vector #6.1 we have: $f_1(00010) = 0^*$ (as given by the expert) and thus we can infer (by utilizing monotony) that $f_1(00000)=0$, where (00000) is vector #10.1. The values in column 3 for f_1 are derived by *"up-down sliding"* in table 1 according to the following five steps:

Step 1. Begin from the first vector #1.1 (i.e., (01100)).
Action: Ask the expert about the value of $f_1(01100)$.
Result: The expert reported that: $f_1(01100)=1$.

Step 2. Write "1^*" next to vector #1.1 (and under column 3). Recall that an asterisk denotes an answer directly provided by the expert. Apparently, if the reply were false (0), then we had to write "0^*" in column 3. The case of having a false value corresponds to column 7 in table 1.

Step 3. **Action:** Go to column 6 to find vectors to extend the true (1) value.
Result: Vectors #1.2, #6.3, and #7.3.
Comment: They should yield (because of monotonicity) true value (1).
Therefore, in column 3 and next to examples #1.2, #6.3, and #7.3 the values of the function must be (1) (note that now no asterisk is used).

Step 4. **Action:** Go to vector #1.2 (next vector while sliding down). Check whether the value of $f_1(11100)$ has already been fixed or not.
If the value of $f_1(11100)$ is not fixed (i.e., it is empty), then repeat steps 1-3, above, for this new

vector (i.e., vector #1.2).
If $f_1(11100)$ is not empty (i.e., it has been already fixed), then go to the next vector while sliding down (i.e., move to vector #2.1).
(Note that if the value has not been fixed yet, then we will denote it by $f_i(x) = e$; for *e*mpty.)
Result: $f_1(11100) \neq e$. Extent the values of the function for the vectors #6.4 and #7.4. and go to the next vector in the table (i.e., move to vector #2.1).

Step 5. The next vector is #2.1. Continue as above with step 4 but now we have vector #2.1 (instead of vector #1.2).

The above procedure is repeated until all vectors and functions have been covered. The interpretation of what happens in columns 8 to 11 is provided in the next sub-section. In order one to construct formula (5-3) (which was shown in section 5.2) one needs to concentrate in the information depicted in columns 2 and 3 in table 1. One needs to take the first vector marked with "1*" in each one of the chains and construct for each of these vectors a conjunction of non-zero components. For instance, for the vector (01010) in chain 2 the corresponding conjunction is x_2x_4. Similarly, from chain 6 we have taken the "1" components in the vector (00110) and formed the conjunction x_3x_4.

We obtained the Boolean expressions (5-2) for $x_2 = \psi(y_1,y_2,y_3,y_4,y_5)$ from the information depicted in table 1 (columns 2 and 5) with the following steps: *(i)* First find all the maximal lower units for all chains as elementary conjunctions; *(ii)* exclude the redundant terms (conjunctions) from the end formula. Thus, at first from table 1 (columns 2,5) we obtained $x_2 = y_1y_2 \vee y_2y_3 \vee y_2y_4 \vee y_1y_3 \vee y_1y_4 \vee y_2y_3y_4 \vee y_2y_3y_5 \vee y_2 \vee y_1 \vee y_3y_4y_5$ and then we simplified it to $y_2 \vee y_1 \vee y_3y_4y_5$.

Similarly as above, from columns (2,3,4) in table 1 we obtained the initial components of the target functions of x_1, x_2, x_3, x_4, x_5 for the **biopsy** sub-problem as follows:

$$f_1(x) = x_2x_3 \vee x_2x_4 \vee x_1x_2 \vee x_1x_4 \vee x_1x_3 \vee x_3x_4 \vee x_3 \vee x_2x_5 \vee x_1x_5 \vee x_5,$$

and for the **cancer** sub-problem to be defined as:

$$f_2(x) = x_2x_3 \vee x_1x_2x_4 \vee x_1x_2 \vee x_1x_3x_4 \vee x_1x_3 \vee x_3x_4 \vee x_3 \vee x_2x_5 \vee x_1x_5 \vee x_4x_5.$$

The simplification of these disjunctive normal form (DNF) expressions allowed us to exclude some redundant conjunctions. For instance, in x_2 the term Y_1Y_4 is not necessary, because y_1 covers it. Thus, the right hand side parts in expressions (5-1) to (5-4) form the minimal DNFs.

5.4. Sequential Search for Nested Functions

Next we show how it is possible to further decrease the number of calls using a modification of the algorithm described for solving the previous main problem. Recall that the functions F_1 and F_2 are nested because $E_2^+ \subseteq E_1^+$. That is, for any input example X the following is true: $F_2(X) \leq F_1(X)$. The last statement means that if we recognize cancer, then we recognize the necessity of biopsy too. The property $F_2(X) \leq F_1(X)$ means that:

$$\text{if } F_2(X)=1, \text{ then } F_1(X)=1, \tag{5-7}$$

and

$$\text{if } F_1(X)=0, \text{ then } F_2(X)=0. \tag{5-8}$$

The above realization permits to avoid calls to the expert for determining the right hand sides of (5-7) and (5-8), if one knows the values of the left hand side.

The above idea can be used to make the dialogue with the expert more efficient. This was done in this experiment and is described in table 1. Column 8 in table 1 shows the results of using the values of F_2 (column 4) and property (5-7) to restore function F_1. For instance, for vector #6.2 according to the expert we have $F_2(00110)=1$. Hence, by using (5-7) we should also have $F_1(00110)=1$ **without** an additional call to the expert. Column 9 represents the same situation for F_2, if one knows the expert's answers for F_1. In this case we use the pair {relation (5-8), column (4)} instead of the pair {relation (5-7), column 5}, as was the case before.

Note that the asterisks "*" in columns 8 and 9 show the necessarily needed calls. In order to restore function F_1 by using information regarding values of F_2 we asked the expert 7 times, instead of the 13 calls we had to use for the independent restoration of function F_1. That is, now we were able to use about 2 times less calls. In particular, the value $F_1(10001)=1$ for vector #9.2 was inferred from the fact $F_2(10001)=1$. However,

to restore function F_2 by using F_1 we asked the expert 13 times. That is, we asked him as many times as during the independent restoration of F_2 (i.e., the nested approach was not beneficial in this case). This should not come as surprise because the limits described in the previous sections are upper limits. That is, on the average the sequential search is expected to outperform a non-sequential approach, but cases like the last one can still be expected.

The previous analysis shows that, in this particular application, the most effective way was at first to restore F_2 with 13 calls and next to use this function values to restore F_1, which required only 7 additional calls. Restoration of F_2 by using information of values of function F_1 required 13 calls, and to restore both functions in E_5 would had required 13 + 13 = 26 calls. In the sequence of the restoration $<F_2, F_1>$ the total amount of calls to restore both functions is 13 + 7 = 20 calls in comparison with 13 + 13=26 calls for independent restoration. We should also add to both cases 12 + 4 calls which were needed to restore functions φ and ψ. Therefore, in total we needed 20 + 16 = 36 and 26 + 16 = 42 calls, respectively. Note that we began from $2\times2{,}048$ and 2×924 calls for both functions. Our totals of 36 and 42 calls are about 100 times less than the number of non optimized calls (i.e., $2\times2{,}048$) and about 50 times less than the upper limit guaranteed according to Hansel lemma (i.e., the 2×924 calls).

5.5. A Joint Search Approach for Nested Functions

Next we study the possibility to decrease the number of calls once more with a joint search approach of F_1 and F_2 by using the following switching strategy:

Step 1: Ask the expert for the value of $F_2(X^{1.1})$ for vector #1.1.

Step 2: If $F_2(X^{1.1})=1$, then ask for the first vector of the next chain. That is, ask for the value of $F_2(X^{2.1})$ for vector #2.1.
Otherwise, ask for the value of $F_1(X^{1.2})$.

Step 3: If $F_1(X^{1.1})=0$, then ask for the value of $F_1(X^{1.2})$ for vector #1.2.
Otherwise, switch to ask for the value of $F_2(X^{1.1})$.

The generalization of the previous steps for arbitrary $X^{I.K}$ is done with steps A and B. Step **A** is for F_1 and step **B** for F_2. These steps are best described as follows:

Step A: If $F_1(X^{I.K})=0$ and vector #$(i.k)$ is not the last one in the current chain and $F_1(X^{I,K+1}) = E$ (i.e., empty), then ask the expert for the value of $F_1(X^{I,K+1})$. Otherwise, ask for the value for the first vector $X^{I+1.J}$ from the next chain such that $F_1(X^{I+1.J}) = E$.
If $F_1(X^{I.K}) = 1$ and $F_2(X^{I,K}) = E$, then ask for the value of $F_2(X^{I,K})$.
If $F_1(X^{I.K}) = 1$ and $F_2(X^{I,K}) = 0$, then ask for the value of $F_2(Y)$, where Y is the first vector from the same or the next chain such that $F_2(Y) = E$.

Step B: If $F_2(X^{I.K}) = 1$, then ask for the first vector Y of the next chain such that $F_2(Y) = E$.
If $F_2(X^{I.K}) = 0$ and $F_1(X^{I.K}) = E$, then ask the expert for the first vector y such that $F_1(Y) = E$.

The results of applying this strategy to restore the two functions F_1 and F_2 are presented in table 1 (in columns 10 and 11, respectively). The number of calls to restore both F_1 and F_2 is equal to 22 (see table 1, columns 10 and 11) by asking for the values of F_1 8 times and for the values of F_2 14 times (see also the values marked with "*" in columns 10 and 11). Note that the previous algorithm required 22 calls.

TABLE 2
Comparison of the results in E_5 and E_3

#	ways of search	F_1	F_2	F_1, F_2	index1	index2	φ	ψ
1	2	3	4	5	6	7	8	9
1	Non-optimized search (upper limit)	32	32	64	1	__	8	32
2	Optimal search (upper limit)	20	20	40	1.6	1	6	20
3	Independent search	13	13	26	2.5	1.5	4	12
4	Sequential search $<F_1, F_2>$	13	13	26	2.5	1.5		
5	Sequential search $<F_2, F_1>$	7	13	20	3.2	2.0		
6	Joint search	__	__	22	2.9	1.8		

TABLE 3
Comparison of the results in E_{11}

#	ways of search	F_1	F_2	F_1, F_2	index1	index 2
1	2	3	4	5	6	7
1	Non-optimized search (upper limit)	2,048	2,048	4,096	1	__
2	Optimal search (upper limit)	924	924	1,848	2.22	1
3	Independent search	29	29	42	97.5	48.8
4	Sequential search $<F_1, F_2>$	29	13	42	97.5	48.8
5	Sequential search $<F_2, F_1>$	7	29	36	113.8	51.3
6	Joint search	__	__	38	107.8	48.6

Next in tables 2 and 3 we summarize the various results for interviewing the expert under the different strategies. Table 2 represents numbers of calls for the different kinds of search for the functions F_1, F_2, ψ in E_5 and for φ in E_3. Table 3 represents results for E_{11}. These results summarize the numbers of calls needed for the full restoration of F_1 and F_2 as functions of 11 binary variables at level 2. In particular, note that the independent search of F_1 required 13 calls in E_5, 12 calls for ψ and 4 calls for φ. That is, the 29 calls shown in table 3. The sequential search $<F_2, F_1>$ required the same number of calls for F_2, but only 7 calls for F_1, because we used the same φ and ψ functions found for F_1 (see also tables 2 and 3).

The total amount shown in column "F_1, F_2" (i.e., under column 5) represents the number of calls to restore both functions. For the last four hierarchical searches in this column we excluded the non necessary calls for the second restoration of ψ and φ, i.e., the 16 calls. For instance, independent search required 29 calls to restore both F_1 and F_2 in E_5. That is, a total of 42 calls in E_{11}, as shown in this column.

Next, let us describe the meaning of indexes 1 and 2 in tables 2 and 3. We denote the upper limit of calls for non optimized search as In_1 and the analogous amounts for other ways of the search for both functions (columns 5) as In_i (for i = 2,3,4,5,6). In these terms Index1=In_1/In_i (for i = 2,3,4,5,6) and index2 = In_2/In_i (for i = 3,4,5,6). In particular, index1 shows that we used 3.2 times less calls than In_1 and 2.0 times less calls than In_2 in E_5, and also 113.8 times less than In_1 and 51.3 times less calls than In_2 in E_{11}. These interactive experiments demonstrate the potential for achieving significantly high efficacy of the proposed approach for interactive restoration of monotone Boolean functions.

6. CONCLUDING REMARKS

Some computational experiments (see, for instance, [9] and [26]) have shown that it is possible to significantly decrease the number of questions to an oracle in comparison with the full number of questions (which is equal to 2^n) and also in comparison with a guaranteed pessimistic estimation (formula (2-2) in section 2) for many functions. Some close form results were also obtained for a connected problem of retrieval of the maximal upper zero [18]. The results in this paper demonstrate that an interactive approach, based on monotone and hierarchical Boolean functions, has the potential to be very beneficial to interactive machine learning.

Acknowledgements

The authors are very grateful to Dr. James F. Ruiz, from the WOMAN'S HOSPITAL of Baton Rouge, LA, for his expert assistance in formulating the medical example described in this paper.

REFERENCES

1. D. Angluin, Queries and concept learning, *Machine Learning* 2(4):319-342 (1988).
2. V.B. Alekseev, Monotone Boolean functions, *Encyclopedia of Mathematics*, Vol. 6, Kluwer Academic Publishers, 1988, pp. 306-307.
3. M. Bongard, PATTERN RECOGNITION, Moscow, "Nauka" Publications (in Russian), 1967, (English translation in 1970, by Spartakos Press, New York, NY, USA).
4. E. Boros, P.L. Hammer and T. Ibaraki, Predicting cause-effect relationships from incomplete discrete observations, *SIAM Journal on Discrete Mathematics* 7(4):531-543 (1994).
5. D. Cohn, L. Atlas and R. Ladner, Improving generalizing with active learning, *Machine Learning* 15:201-221 (1994).
6. R. Dedekind, Ueber Zerlegungen von Zahlen durch ihre grossten gemeinsamen Teiler, *Festschrift Hoch. Braunschweig* (in German) 1897, u.ges. Werke, II. pp. 103-148.
7. L.M. Fu, Knowledge-based connectionism for revising domain theories, *IEEE Transactions on Systems, Man, and Cybernetics* 23(1):173-182 (1993).
8. S. Goldman and R.H. Sloan, The power of self-directed learning, *Machine Learning* 14:271-294 (1992).
9. Y. Gorbunov and B. Kovalerchuk, An interactive method of monotone Boolean function restoration, *Journal of Academy of Science of UzSSR, Engineering* (in Russian) 2:3-6 (1982).
10. P.L. Hammer and E. Boros, Logical analysis: An overview, *RUTCOR Research Report*, Rutgers University, NJ, 1994.
11. G. Hansel, Sur le nombre des fonctions Boolenes monotones den variables, *C.R. Acad. Sci. Paris* (in French), 262(20):1088-1090 (1966).
12. K. Hattoru and Y. Torri, Effective algorithms for the nearest neighbor method in the clustering problem, *Pattern Recognition* 26(5):741-746 (1993).
13. A.P. Kamath, N.K. Karmakar, K.G. Ramakrishnan and M.G.C. Resende, A continuous approach to inductive inference, *Mathematical Programming* 57:215-238 (1992).
14. A.P. Kamath, N.K. Karmakar, K.G. Ramakrishnan and M.G.C. Resende, An interior point approach to Boolean vector synthesis, *Proceedings of the 36-th MSCAS* 1-5 (1994).
15. B. Kamgar-Parsi and L.N. Kanal, An improved branch-and-bound algorithm for computing *k*-nearest neighbors, *Pattern Recognition Letters* 3:7-12 (1985).
16. D. Kleitman, On Dedekind's problem: The number of monotone Boolean functions, *5-th Proceedings of the American Mathematics Society* 21:677-682 (1969).
17. V.K. Korobkov, On monotone Boolean functions of algebraic logic, *Problemy Cybernetiki*, "Nauka"

Publications, Moscow, (in Russian) 13:5-28 (1965).

18. B. Kovalerchuk and V. Lavkov, Retrieval of the maximum upper zero for minimizing the number of attributes in regression analysis, *USSR Computational Mathematics and Mathematical Physics* 24(4):170-175 (1984).

19. B. Kovalerchuk, E. Triantaphyllou and E. Vityaev, Monotone Boolean functions learning techniques integrated with user interaction, *Proceedings of the Workshop "Learning from examples vs. programming by demonstration", 12-th International Conference on Machine Learning*, Tahoe City, CA, USA, (July 1995), 41-48.

20. B. Kovalerchuk, E. Triantaphyllou and J.F. Ruiz, Monotonicity and logical analysis of data: A mechanism of evaluation of mammographic and clinical Data, *Technical Report*, Louisiana State University, Dept. of Industrial Engineering, Baton Rouge, LA 70803-6409, USA, 1995.

21. T. Kurita, An efficient agglomerative clustering algorithm using a heap, *Pattern Recognition* 24(3):205-209 (1991).

22. O.L. Mangasarian, W.N. Street and W.H. Woldberg, Breast Cancer Diagnosis and Prognosis Via Linear Programming, *Operations Research* 43(4):570-577 (1995).

23. J.W. Shavlik, Combining symbolic and neural learning, *Machine Learning* 14:321-331 (1994).

24. E. Triantaphyllou, A.L. Soyster and S.R.T. Kumara, Generating logical expressions from positive and negative examples via a branch-and-bound approach, *Computers and Operations Research* 21(2):185-197 (1994).

25. E. Triantaphyllou, Inference of a minimum size Boolean function from examples by using a new efficient branch-and-bound approach, *Journal of Global Optimization* 5(1):69-94 (1994).

26. E. Triantaphyllou and A.L. Soyster, An approach to guided learning of Boolean functions, *Mathematical and Computer Modeling* 23(3):69-86 (1996).

27. V.N. Vapnik, *Estimating of Dependencies Based on Empirical Data*, Springer-Verlag, New York, NY, 1982.

28. L. Valiant, A theory of learnable, *Communications of the ACM* 27:1134-1142 (1984).

29. E. Vityaev and A. Moskvitin, Introduction to discovery theory and the DISCOVERY system, Logical Methods in Informatics, *Computational Systems (Vychislitel'nye sistemy)* Institute of Mathematics, Russian Academy of Science, Novosibirsk, (in Russian) 148:117-163 (1993).

Received 30 November 1995; revised 10 February 1996