For reference information, please see the last page.

9

SOME RECENT DEVELOPMENTS OF USING LOGICAL ANALYSIS FOR INFERRING A BOOLEAN FUNCTION WITH FEW CLAUSES

Evangelos Triantaphyllou
Boris Kovalerchuk
and Aniruddha S. Deshpande

ABSTRACT

One of the main areas of great potential in the interface of operations research and computer science is in inductive inference. Inductive inference refers to the extraction of a pattern from observations which belong to different classes. This is the essence of building many intelligent systems with learning capabilities. In this paper we discuss a logical analysis approach to this problem. Given are sets of binary vectors. The main problem is how to extract a Boolean function, in CNF or DNF form, with as few clauses (conjunctions or disjunctions) as possible. Therefore, this is an optimization problem. We present a number of theoretical results and also some possible extensions.

1 INTRODUCTION

A common problem in many scientific and engineering applications is the identification of the characteristics or combination of characteristics which can explain the occurrence of some desirable or undesirable effects. Traditionally, such challenges are examined by statistical and/or artificial intelligence procedures. Among the specific procedures used are: clustering, factor analysis, discriminant analysis, linear regression, decision trees, and neural networks. All this kind of approaches, however, are limited because they rely on some assumptions which are probabilistic in nature (e.g., regarding the existence of a statistical distribution on the data) or geometric in nature (e.g., the existence of separating hyperplanes in some related Euclidean space) [Boros *et al.*, 1994]. These assumptions may severely limit the applicability of the above methods on some real life data.

Moreover, very often the output of these procedures is difficult to be interpreted by field experts. For instance, medical doctors may feel uncomfortable in accepting diagnoses made by a neural network. This situation is increasingly recognized lately and some hybrid systems try to combine rule based systems with neural network techniques [Fu, 1991]. Such hybrid systems produce a set of logical rules which in turn can be used for the classification of new observations. However, the results ares not encouraging because some recent research [Shavlik, 1994] shows that the number of rules generated in this manner can be exponentially large.

2 THE LOGICAL ANALYSIS APPROACH

Logical analysis can be used to extract a set of logical rules (decision rules) which describe patterns of classes of observations. The central hypothesis of using logical analysis [Boros *et al.*, 1994] is that *many phenomena are governed by logical decision rules*. Thus, logical analysis aims at discovering these rules. Logical analysis attempts to capture the cause-effect relationships contained in a data set.

In a typical problem available are collections of examples. Each example is described in terms of a finite set of attributes (also called atoms, variables, characteristics, parameters, or factors). These examples are classified into a finite set of mutually exclusive and exhaustive classes (usually only two classes are considered). Then, the problem is to extract a set of logical rules which describe the patterns implied by these data. The main steps required when applying this kind of data analysis are as follows:

- Definition of the relevant characteristics (i.e., parameters or attributes or factors).
- Extraction of the patterns (i.e., the logical decision rules) which can explain the observations in the various classes.

The above concepts are best described in the following illustrative example.

Suppose that the functioning behavior of a system depends on the values of the five Boolean variables: A_1, A_2, A_3, A_4, and A_5. That is, each such parameter takes on a value which is either true (denoted by 1) or false (denoted by 0). Suppose that the following observations in the set E^+ describe some examples in which the system was functioning correctly, while the observations in the set E^- describe some examples in which the system was malfunctioning:

$$E^+ = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad E^- = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For instance, the first example in E^+ (i.e., [0 1 0 0 0 1 0 1 1 1]) implies that when the values of (A_1, A_2, A_3, A_4, *and* A_5) are equal to (*FALSE, TRUE, FALSE, FALSE, FALSE*), respectively, (or equivalently, the values of ($\bar{A}_1$, $\bar{A}_2$, $\bar{A}_3$, $\bar{A}_4$, *and* $\bar{A}_5$) are equal to (*TRUE, FALSE, TRUE, TRUE, TRUE*), respectively), then the system functions correctly (because this example belongs to E^+). The main problem is to use these two classes of observations (or examples of behavior) and describe the conditions on the values of the attributes which cause these examples to belong to either one of these two classes (sets with observations).

Next, consider the following Boolean function defined on the previous five Boolean parameters (or their negations) given in CNF form:

$$(A_1 \vee A_2 \vee A_3) \wedge (\bar{A}_1 \vee A_2 \vee \bar{A}_3 \vee \bar{A}_4) \wedge (\bar{A}_1 \vee A_3 \vee \bar{A}_5) \,.$$

It can be easily verified that the above expression satisfies the requirements of the previous examples. That is, each positive example makes *each* conjunction to have true value. Furthermore, each negative example is rejected by at *least one* of the conjunctions. In other words, the previous Boolean function returns true value for each one of the positive examples, while it returns false value for each one of the negative examples.

Given the previous E^+ and E^- data, then a new and unclassified example is a positive example (i.e., describes a correct behavior of the system under consideration) if all the previous three rules are satisfied. Otherwise, the example will be classified as a negative one. For the above reasons the terms "*Boolean function*" and "*logical decision rules*" will be used in this paper to mean the same concept.

3. LITERATURE REVIEW

The extraction of a set of logical decision rules from collections of classified data is a type of learning from examples. Complexity issues of this type of learning can be found in [Valiant, 1984 and 1985], [Kearns *et al.*, 1987], and [Pitt and Valiant, 1988]. A considerable amount of related research is today known as the PAC (Probably Approximately Correct) learning theory (see, for instance, [Angluin, 1988] and [Haussler and Warmuth, 1993]). Since the very early days it was recognized that the problem of inferring a Boolean function with a specified number of clauses is NP-complete (see, for instance, [Brayton *et al.*, 1985] and [Gimpel, 1965]).

A learning algorithm is a computational procedure which takes a random sample of positive and negative examples of the target concept c and returns a hypothesis h. In the literature a learning algorithm A is a PAC algorithm if for all positive numbers ε and δ (where: $1 > \varepsilon, \delta > 0$), when A runs and accesses unclassified examples, then it eventually halts and outputs a concept h with probability at least $1 - \delta$ and error at most ε [Angluin, 1992]. A related issue is the one of *properly PAC learnable* concepts. This occurs when the concept c and the hypothesis h are assumed to belong to the same spaces of concepts [Haussler and Warmuth, 1993].

Conjunctive concepts are properly PAC learnable [Valiant, 1984]. However, the class of concepts in the form of the disjunction of two conjunctions is not properly PAC learnable [Pitt and Valiant, 1988]. The same is also true for the class of existential conjunctive concepts on structural instance spaces with two objects [Haussler, 1989]. The classes of k-DNF, k-CNF, and k-decision lists are properly PAC learnable for each fixed k [Kearns *et al.*, 1987], but it is unknown whether the classes of all DNF, or CNF functions are PAC learnable [Haussler and Warmuth, 1993] and [Goldman, 1990]. Note that the present paper deals with Boolean functions in CNF or DNF form. Therefore, the learning algorithms described later in this paper are not known whether they are PAC learnable. Also, in [Mansour, 1992] an $n^{O(\log\log n)}$ algorithm is given for learning DNF formulas (however, *not* of minimal size) under a uniform distribution using membership queries.

There are many reasons why one may be interested in inferring a Boolean function with the minimum (or near minimum) number of terms. In a circuit design environment, a minimum size Boolean representation is the prerequisite for a successful VLSI application. In a learning from examples environment, one may be interested in deriving a *compact set* of decision rules which satisfy the requirements of the input examples [Hammer and Kogan, 1992]. Too many rules may increase the chances for errors. When, for instance, one wishes to use the rules to control the safety of a structure, then a small number of rules will also require a small number of sensors. In medical diagnosis one may wish to derive a small set of rules, because too many rules are difficult to be verified and validated. These are the main reasons why we take an optimization approach in dealing with the problem of inferring a small number of logical decision rules (i.e., a compact Boolean function).

In [Triantaphyllou, Soyster, and Kumara, 1994] the problem of inferring a set of logical clauses is examined. In that work a rather small set of clauses (not necessarily of minimal size) is determined. That approach is based on a branch-and-bound (B&B) algorithm. That B&B algorithm was later significantly enhanced and improved in [Triantaphyllou, 1994].

There is another clause inference approach which can be used to determine a minimal size set of logical clauses. This method, denoted as SAT (for satisfiability), has been proposed in [Kamath *et al.*, 1992]. Then a *clause satisfiability* model is formed as follows. Let M_1 and M_2 be the numbers of examples in the E^+ and E^- sets, respectively. In [Kamath *et al.*, 1992] it is shown that given two collections of positive and negative examples (called the *ON-set* and *OFF-set*, respectively) then, then a DNF system can be inferred to satisfy the requirements of these examples. This is achieved by formulating a satisfiability problem and then using the interior point method developed by Karmakar and his associates [1992] to solve it. This approach pre-assumes the value of *k*; the number of conjunctions in the DNF system. The SAT problem uses the following Boolean variables (Kamath et al. [12]):

if A_i *is in the j-th conjunction*

if A_i *is not in the j-th conjunction*

if $\bar{A}_i$ *is in the j-th conjunction*

if $\bar{A}_i$ *is not in the j-th conjunction*

if $A_i = 1$ *in the positive example* $\alpha \in E^+$

if $A_i = 0$ *in the positive example* $\alpha \in E^+$

if the positive example α *is accepted by the j-th co*

otherwise

Then, the clauses of this SAT problem are as follows:

$$s_{ji} \bigvee s'_{ji}, \quad \text{for } i = 1, \ldots, n, \text{ and } j = 1, \ldots, k, \tag{1}$$

$$\Big(\bigvee_{i \in P_r} \bar{s}'_{ji}\Big) \vee \Big(\bigvee_{i \in \bar{P}_r} \bar{s}_{ji}\Big), \quad \text{for } j = 1, \ldots, k, \text{ and } r = 1, \ldots, M_2, \tag{2}$$

$$\bigvee_{j=1}^{k} z_j^{\alpha}, \quad \text{for } \alpha = 1, \ldots, M_1, \tag{3}$$

$$\sigma_{ji}^{\alpha} \vee \bar{z}_j^{\alpha}, \quad \text{for } i = 1, \ldots, n, \ j = 1, \ldots, k, \text{ and } \alpha = 1, \ldots, M_1, \tag{4}$$

where P_r is the set of indices of A for which $A_i = 1$ in the negative example $r \in E^-$. Similarly, P_r is the set of indices of A for which $A_i = 0$ in the negative example $r \in E^-$.

Clauses of type (1) ensure that never both A_i and A_i will appear in any conjunction. Clauses of type (2) ensure that each negative example is rejected by all conjunctions. Clauses of type (3) ensure that each positive example is accepted by *at least one* conjunction. Finally, clauses of type (4) ensure that $z_j^\alpha = 1$ if and only if the positive example α is accepted by the j-th conjunction. In general, this SAT problem has $k(n(M_1 + 1) + M_2) + M_1$ clauses, and $k(2n + M_1)$ Boolean variables. A detailed example on this formulation can be found in [Kamath *et al.*, 1992].

If the above clause satisfiability problem is feasible, then the conclusion is that it is possible to correctly classify all the examples with k or fewer clauses. If this SAT problem is infeasible, then one must increase k until feasibility is reached. In this manner, the SAT approach can yield a system with the minimum number of clauses. It is very important one to observe at this point that computationally it is much harder to prove that a given SAT problem is infeasible than it is feasible. Therefore, trying to determine a minimum size Boolean function by using the SAT approach may be computationally too difficult.

It should also be emphasized here that it is not very critical whether an inference algorithm determines a CNF or DNF system (i.e., CNF or DNF Boolean function). By applying theorem 1 (described later in section 4.1), either a CNF or DNF system can be derived by using *any algorithm*. Also, some computational results indicate that the B&B approach proposed by the first author in [Triantaphyllou, 1994] is much more efficient than the previous satisfiability based approach.

The SAT results were derived by using a VAX 8700 computer running the 10-th Edition *UNIX*. The code for the SAT tests was written in FORTRAN and in C. The SAT results were originally reported in Kamath *et al.* [1992]. The OCAT results were derived by using an IBM 3090-600S computer running *XA/VM* and the code was written in FORTRAN. On the average, in those tests the OCAT/B&B approach was 5,579 times faster [Triantaphyllou, 1994]. However, it should be stated at this point that the IBM3090-600S machine is, in general, 4-5 times faster than the VAX 8700 computer used in the SAT tests. Thus, due to the different hardware/software platforms and the limited number of test problems, the current results can convey only a flavor on the relative performance of the two methods, and by no means should be considered as a direct comparison of the two approaches.

A related problem is to study the construction of a partially defined Boolean function (or *pdBf*), given disjoint sets of positive and negative examples [Boros *et al.*, 1994]. That is, now it is required that the atoms of the function be grouped according to a given scheme (called a decomposition structure). The problem then is to determine whether a given *pdBf* has an extension when a specific decomposition structure is considered. Typically, a *pdBf* may have exponentially many different extensions. In [Boros *et al.*, 1994] computational and existence issues of this type of problem are examined for many different cases. Note that very often (i.e., depending on the case) these problems are NP-complete. In that treatment Boolean functions were not attempted to be represented by a minimum size CNF or DNF expression (as is the case in this paper).

Other approaches dealing with this type of learning is to use neural networks. However, although a neural network approach may yield encouraging results, it fails to extract the underlying logic in the form of a set of decision rules (i.e., a Boolean function). Some recent efforts have been successful in combining symbolic approaches to neural networks and thus generate rules [Fu, 1994]. However, they are impractical since they may generate an exponential number of rules [Shavlik, 1994]. As result of this, field experts hesitate to use neural network approaches because they do not understand the decision-making process and thus feel uncomfortable.

Mangasarian and his associates [1991] did some considerable work in using linear programming in solving similar classification problems. That approach constructs a family of pairs of separating hyper-planes. Then, new instances are classified by examining where they belong in relation to these planes. They applied their approach in diagnosing breast cancer and also in diagnosing heart disorders. The proposed logical analysis approach, however, besides the promising computational results, it also provides a set of decision rules which can be easily understood by medical doctors. As result, it has the potential to become accepted anywhere other methods have failed to gain the trust of the end users.

4 CURRENT RESEARCH RESULTS

4.1 Some Definitions and Terminology

Let $\{A_1, A_2, A_3, \ldots, A_t\}$ be a set of t Boolean *predicates* or *atoms*. Each atom A_i ($i=1,2,3,\ldots,t$) can be either true (denoted by 1) or false (denoted by 0). Let F be a *Boolean function* over these atoms. That is, F is a mapping from $\{0,1\}^t - \{0,1\}$ which determines for each combination of truth values of the arguments $A_1, A_2, A_3, \ldots, A_t$ of F, whether F is true or false (denoted as 1 or 0, respectively). For each Boolean function F, the *positive examples* are the vectors $v \in \{0,1,\}^t$ such that $F(v) = 1$ (i.e., they are *accepted* by F). Similarly, the *negative examples* are the vectors $v \in \{0,1,\}^t$ such that $F(v) = 0$ (i.e., they are *rejected* by F).

In the previous paragraph a Boolean function F is assumed to be of any form. However, in propositional calculus it is convenient to represent Boolean functions using the conjunctive normal form (CNF) or the disjunctive normal form (DNF). See, for instance, [Blair *et al.*, 1985], [Cavalier *et al.*, 1990], [Hooker, 1988a and 1988b], [Jeroslow, 1988 and 1989], and [Williams, 1986]. Peysakh in [1987] describes an algorithm for converting any Boolean expression into CNF. In this paper a set of positive examples will be denoted as E^+. Similarly, a set of negative examples will be denoted as E^-.

Next, define the general form of a CNF and DNF system as (I) and (II), respectively. That is:

$$\bigwedge_{j=1}^{k} \left(\bigvee_{i \in \rho_j} a_i \right), \tag{I}$$

and

$$\bigvee_{j=1}^{k} \left(\bigwedge_{i \in \rho_j} a_i \right), \tag{II}$$

where a_i is either A_i or $\bar{A}_i$ and ρ_j is the superset of the indices of the atoms in the j-th conjunction or disjunction. Furthermore, $\bar{v}$ is defined as the *complement* of example v. For instance, if $v = [1,0,0]$, then $\bar{v} = [0,1,1]$. The following definition introduces the concept of the complement of a set of examples. Let E be a collection of examples (either positive or negative). Then, $\bar{E}$ is defined as the *complement of the collection E.*

The following theorem (proved in [Triantaphyllou and Soyster, 1995a]) states an important property which exists when CNF and DNF systems are inferred from collections of positive and negative examples. It is important to observe here that this theorem is independent of the algorithm used to infer a Boolean expression from positive and negative examples.

Theorem 1: *Let E^+ and E^- be the sets of positive and negative examples, respectively. A CNF system given as (I) satisfies the constraints of the E^+ and E^- sets if and only if the DNF system given as (II) satisfies the constraints of $\bar{E}^-$ (considered as the positive examples) and $\bar{E}^+$ (considered as the negative examples).*

4.2 The One Clause At a Time Approach

In [Triantaphyllou, Soyster, and Kumara, 1994] an algorithm which infers CNF systems from positive and negative examples is developed. In that approach, CNF clauses are generated in a way which attempts to minimize the number of CNF clauses that constitute the recommended CNF system. In this way, a compact CNF system can be derived. The strategy followed there is called the *One Claüse At a Time* approach.

The One Clause At a Time (or OCAT) approach is greedy in nature. It uses as input data two collections of positive and negative examples. It determines a set of CNF clauses that, when taken together, rejects all the negative examples and accepts all the positive examples. The OCAT approach is sequential. In the first iteration it determines a single clause (conjunction) which accepts all the positive examples in the E^+ set while it rejects as many negative examples in E^- as possible. This is the greedy aspect of the approach. In the second iteration it performs the same task using the original E^+ set but the revised E^- set has only those negative examples which have not been rejected by any clause (i.e., the first clause) so far. The iterations continue until a set of clauses is constructed which reject all the negative examples in the original E^- set. More on this approach can be found in [Triantaphyllou, Soyster, and Kumara, 1994] and [Triantaphyllou, 1994]. Figure 1 summarizes the iterative nature of the OCAT approach. It is interesting to observe here that the LP approach proposed by Mangasarian and his associates [1991] also uses a greedy approach in the way the hyper-planes are determined.

The core of the OCAT approach is step 2, in figure 1. In [Triantaphyllou, Soyster, and Kumara, 1994] a branch-and-bound based algorithm is presented which solves the problem posed in step 2. That algorithm was later significantly enhanced and improved in [Triantaphyllou, 1994]. The OCAT approach returns the set of desired clauses (i.e., the CNF or DNF system) as set C.

A related problem was examined in [Turksen and Zhao, 1993]. In that development it is shown that Quinlan's ID3 algorithm [Quinlan, 1979] is equivalent with the result of applying a Boolean simplification treatment on the rules generated by ID3. Thus, the authors propose a scheme which first derives a set of rules by using the ID3 algorithm, and then this set is simplified (fewer logical clauses are derived) by applying a Boolean simplification procedure. Some experiments contacted by the authors of this paper compared the OCAT approach with the ID3 algorithm in terms of the number of CNF clauses derived. It was shown that the OCAT approach, when it was combined with the heuristic described in [Deshpande and Triantaphyllou, 1996] created by far much less clauses. The logic inference procedures developed by Triantaphyllou and his associates directly aim at deriving a small number of clauses from collections of positive and negative examples.

The procedure proposed by Turksen and Zhao applies a two step approach: In the first step a system is induced by using the ID3 algorithm and in the second step that system is reduced to an equivalent system with (hopefully) fewer terms. That is, the result of the Turksen/Zhao approach may be drastically dependent by the outcome of the ID3 algorithm. However, no experiment results are available to compare the Turksen/Zhao approach with the OCAT approach.

4.3 The Rejectability Graph of Two Collections of Examples

This section presents the motivation and definition of a special graph which can be easily derived from positive and negative examples. The concept of this graph was originally developed in [Triantaphyllou and Soyster, 1994].

```
$i = 0$ ; $C = \phi$;
DO WHILE ($E^- \neq \phi$)
    Step 1: $i \leftarrow i + 1$;  /* $i$ stands for the $i$-th clause */
    Step 2: Find a clause $c_i$ which accepts all members of $E^+$
            while it rejects as many members of $E^-$ as possible;
    Step 3: Let $E^-(c_i)$ be the set of members of $E^-$ which are rejected by $c_i$;
    Step 4: Let $C \leftarrow C \cup c_i$;
    Step 5: Let $E^- \leftarrow E^- - E^-(c_i)$;
REPEAT;
```

Figure 1. The One Clause At a Time (OCAT) Approach.

To see the motivation for introducing this graph consider a situation with $t = 5$ atoms. Suppose that the vector $v_1 = [1,0,1,0,1]$ is a positive example while the two vectors $v_2 = [1,0,1,1,1]$ and $v_3 = [1,1,1,0,1]$ are negative examples. If $\bar{A}_i$ denotes the negation of the atom A_i, then the positive example v_1 indicates that A_1, $\bar{A}_2$, A_3, A_4, and A_5 are true (or equivalently, $\bar{A}_1$, A_2, $\bar{A}_3$, A_4 and $\bar{A}_5$ are false in this example). Similar interpretations can easily be derived for the remaining two examples v_2 and v_3.

Let the set $T(v)$ denote the set of the atoms that are true in the example (either positive or negative) v (where $v \in \{1,0,\}^t$, and t is the number of atoms). When this definition of the $T(v)$ set is applied on the previous three examples v_1, v_2, and v_3, then the following sets are derived:

$$T(v_1) = T(\,[1,0,1,0,1]\,) = \{A_1, \bar{A}_2, A_3, \bar{A}_4, A_5\}$$
$$T(v_2) = T(\,[1,0,1,1,1]\,) = \{A_1, \bar{A}_2, A_3, A_4, A_5\}$$
$$T(v_3) = T(\,[1,1,1,0,1]\,) = \{A_1, A_2, A_3, \bar{A}_4, A_5\}.$$

Consider a single CNF clause (conjunction), denoted as C, of the form (where a_i is either A_i or $\bar{A}_i$, for some $N \leq t$): $\bigvee_{i=1}^{N} a_i$. Then, the clause C accepts an example v (i.e., v is a positive example of C) if and only if any of the atoms in the set $T(v)$ is one of the atoms involved in the expression $\bigvee_{i=1}^{N} a_i$. Otherwise, the example v is not accepted (i.e., v is a negative example of C). For instance, if the clause C is defined as: $C = (\bar{A}_2 \vee A_4)$, then the examples v_1 and v_2 are accepted by C, while the example v_3 is not accepted.

Using the previous three sets $T(v_1)$, $T(v_2)$, and $T(v_3)$ and the notion of acceptance it can be easily verified that there is no single CNF clause which can simultaneously reject both the two negative examples v_2 and v_3, while at the same time, it accepts the positive example v_1. This is true because any clause which simultaneously rejects the two examples v_2 and v_3, should not involve any of the atoms present in the union of the two sets $T(v_2)$ and $T(v_3)$. But if none of the atoms of the set $\{A_1, A_2, \bar{A}_2, A_3, A_4, \bar{A}_4, A_5\}$ $(= T(v_2) \cup T(v_3))$ is present in the clause, then it is impossible to accept the positive example $v_1 = [1,0,1,0,1]$.

Therefore, given the positive example v_1, then the previous two negative examples v_2 and v_3 cannot be rejected by any single CNF clause (conjunction) which also accepts the positive example v_1. In general, given a set of positive examples E^+, then two negative examples v_1 and v_2 are rejectable by a single CNF clause if and only if the condition in the following theorem [Triantaphyllou and Soyster, 1996a] is satisfied:

Theorem 2: *Let E^+ be a set of positive examples and v_1, v_2 are two negative examples. Let $T(v)$ be the set of the atoms that are true in the (either positive or negative) example v. Then, the two examples v_1 and v_2 are rejectable by a single clause which accepts all the positive examples in E^+ if and only if the following condition is true:*

$$T(v_i) \not\subseteq T(v_1) \cup T(v_2) \text{ for any positive example } v_i \in E^+.$$

Given two collections of positive and negative examples, denoted as E^+ and E^-, respectively, the above theorem motivates the construction of a graph $G = (V, E)$ as follows:

$V = \{ V_1, V_2, V_3, \ldots, V_{M_2} \}$, *where M_2 is the size of the E^- set, and*

$E = \{ (V_i, V_j)$ *if and only if the i-th and the j-th examples in E^- are rejectable by a single clause (subject to the examples in E^+)* $\}$.

We call this graph the *rejectability graph* (or *R*-graph) of E^+ and E^-. Since the previous theorem indicates that it is computationally very easy to examine whether there is an edge between any two vertices of G, the rejectability graph G can be constructed by performing $M_2(M_2-1)/2$ simple rejectability examinations (where

M_2 is the size of the E^- set).

4.4 Properties of the Rejectability Graph

The rejectability graph G of a set of positive and a set of negative examples has a number of interesting properties. To illustrate these properties it is first necessary to consider the following theorem [Triantaphyllou and Soyster, 1996a]:

Theorem 3: *Suppose that the two sets E^+ and E^- are given and F is a family of K negative examples from E^- ($K \leq$ size of set E^-) which are rejected by a single clause subject to the positive examples in E^+. Then, the vertices which correspond to the K negative examples in the rejectability graph G, form a clique in G of size K.*

Although the previous theorem states that any set of negative examples which is rejected by a single clause corresponds to a clique in the rejectability graph, the inverse is not always true. That is, not any clique in the rejectability graph corresponds to a set of negative examples which are rejected by the same clause. In [Triantaphyllou and Soyster, 1996a] the previous statement is demonstrated via a counter-example.

4.4.1 On The Connected Components of The Rejectability Graph.

Another useful application of the rejectability graph G is based on the *connected components* of the graph G. First, observe that any subset of negative examples in the E^- set which is rejected by a single clause, subject to the examples in E^+, corresponds to a subset of vertices of the rejectability graph G which belong to the same connected component of the graph G.

The previous property may become critical when the sets of positive and negative examples are very large. In this case it is useful to first form the rejectability graph G (which is computationally a simple task). Next, determine all the connected components of the rejectability graph by applying an algorithm for finding the connected components. Then, one may solve the smaller clause inference problems which are formed by considering all the positive examples and the negative examples which correspond to the vertices of the connected components in G.

In [Pardalos and Rentala, 1990] there is an excellent survey of algorithms which determine the connected components of a graph. Once the connected components of the rejectability graph G have been found, clauses which reject negative examples, while they also accept all the positive examples, can be determined by considering only one connected component at a time. This approach clearly takes advantage of the connectivity structure of the G graph and provides the means for solving very large clause inference problems.

4.4.2 On The Minimum Clique Cover of The Rejectability Graph.

Consider two sets of negative and positive examples, denoted as E^+ and E^-, respectively. Suppose that C is a *family* of clauses which satisfy the constraints of the set of positive examples E^+ and the set of negative examples E^-. The following two theorems were proved in [Triantaphyllou and Soyster, 1996a] and are very critical in the sense that they establish lower bounds on the minimum number of clauses which reject all the members in E^-, while they accept all the members in E^+. It is important to observe that these theorems are independent of the inference algorithm used to process the positive and negative examples.

Theorem 4: *Suppose that E^+ and E^- are the sets of the positive and negative examples, respectively. Then, C_{min}, the minimum number of clauses which reject all the examples in E^-, while they accept all the examples in E^+, has as a lower bound $\omega(\overline{G})$, the size of the maximum clique of the graph G. Where $\overline{G}$ is the complement of the rejectability graph G of E^+ and E^-. Moreover, the following relation is always true: $C_{min} \geq \omega(\overline{G}) = \alpha(G)$, where $\alpha(G)$ is the* ***stability number*** *of the rejectability graph G.*

In Carraghan and Pardalos [1990] a survey of algorithms which find the maximum clique in any graph is presented. They also present a very efficient algorithm which uses a partial enumeration and outperforms any other known algorithm. In that treatment random problems with 3,000 vertices and over one million edges were solved in rather short times (less than one hour on an IBM ES/3090-900E computer). In any graph G the *chromatic number* $\chi(G)$ is always equal to the clique cover number of $\overline{G}$, (i.e., $k(\overline{G})$). Furthermore, the following is always true:

$$C_{min} \geq k(G) = \chi(\overline{G}) \geq \omega(\overline{G}) = \alpha(G).$$

Therefore, the following theorem [Triantaphyllou and Soyster, 1996a] is true:

Theorem 5: *Suppose that E^+ and E^- are the sets of the positive and negative examples, respectively. Then, C_{min}, the minimum number of clauses which reject all the examples in E^-, while they accept all the examples in E^+, has as a low bound $\chi(\overline{G})$, the chromatic number of the graph G. Where $\overline{G}$ is the complement of the rejectability graph G of E^+ and E^-. Furthermore, this new bound is tighter than the bound given in the previous theorem. That is, the following relation is always true:*

$$C_{min} \geq \chi(\overline{G}) = k(G) \geq \omega(\overline{G}) = \alpha(G).$$

The previous theoretical results can be utilized to decompose large scale problems as follows. First, the connected components of the rejectability graph are determined and the Boolean function construction problem is solved within each connected component. The second approach is also motivated by partitioning the vertices of the rejectability graph into mutually disjoint sets. However, in this second approach, vertices are subdivided via a sequential construction of cliques.

First, the maximum clique of the rejectability graph is determined. The negative examples which correspond to the vertices of the maximum clique, along with *all* the positive examples, form the first sub-problem of this decomposition. Next, the maximum clique of the remaining graph is derived. The second sub-problem is formed by the negative examples which correspond to the vertices of the second clique and all the positive examples. This process continues until all the negative examples (or, equivalently, all the vertices in the rejectability graph) are considered.

We note that this sequence of cliques does not necessarily correspond to a minimum clique cover of the rejectability graph. This procedure is simply a greedy approach which approximates a minimum clique cover. Furthermore, it is possible that a single sub-problem (in which all the vertices in the rejectability graph form a clique) may yield more than one clause.

It should be noted at this point that the clique cover derived by using the above greedy approach may not always correspond to a minimum clique cover. Therefore, the number of cliques derived in that way, cannot be used as a lower bound on the number of clauses derivable from positive and negative examples. Obviously, if the number of cliques is equal to $\omega(G)$, then the previous clique cover is minimal. However, even if the previous clique cover is not of minimum size, it can still be very useful as it can lead to a decomposition of the original problem into a sequence of smaller problems. Some computational results described in [Triantaphyllou and Soyster, 1996a] provide some insight into the effectiveness of such a decomposition approach.

The two problem decomposition approaches described in this section can be combined into one approach. One first decomposes the original problem in terms of its connected components. Next, a clique cover, as described above, is derived for the individual problems which correspond to the connected components of the rejectability graph.

4.5 Clause Inference with Guided Input

The problem of inferring general CNF/DNF clauses with guided input was examined in [Triantaphyllou and Soyster, 1996b]. Suppose that the user can supply the expert with additional examples for correct classification. Then, the *problem of inference with guided input* is how to generate the next example. One obvious approach is to generate the next example randomly. However, this may result in generating many examples and still not achieving a good approximation of the unknown system. It is obviously desirable to consider a sequence of new examples which can lead to a good approximation of the unknown system as quickly as possible.

When a new example is considered, it is given to the expert for the correct classification. Two situations can occur. First, the current system (which is attempting to represent the unknown *"hidden logic"*) classifies the new example in a manner *identical* with the expert (who always correctly classifies each example). In the second case, the new example is classified in the *opposite* way by the expert and the current system. If the current system is not yet a good approximation of the *"hidden logic"*, then the last case is the most desirable scenario. This is true, because in this case one can re-execute a clause inference algorithm (for instance, the OCAT approach) again and, hopefully, derive a closer approximation of the unknown system.

If the current version of the Boolean function is an inaccurate approximation of the *"hidden logic"* and one generates new examples which fail to reveal any contradictions, then additional costs are incurred in classifying new examples, but no improvement is gained. Clearly, it is desirable to use a strategy for determining the next example, such that any possible contradiction between the current version of the Boolean function and the *"hidden logic"* will surface early in the interviewing process.

Next, consider two sets of positive and negative examples, E^+ and E^-, defined on t atoms. Let S_{SAMPLE} denote a logic system (Boolean function) that correctly classifies the sample data, i.e. the examples in E^+ are classified as positive and the examples in E^- are classified as negative (one such function can be obtained via the methods described in [Kamath *et al.*, 1992], [Triantaphyllou *et al.*, 1994], and [Triantaphyllou, 1994]). Also, define S_{HIDDEN} as the *"hidden logic"* Boolean function and $\bar{S}_{HIDDEN}$ as the complement of S_{HIDDEN}. The strategy proposed in [Triantaphyllou and Soyster, 1996b] is based on the following theorem:

Theorem 5: *Suppose that there exists an example $v \in \{0,1\}^t$ such that:*

$$S_{SAMPLE}(v) + S_{R\text{-}SAMPLE}(v) = 0 \quad \text{or:} \qquad (5.a)$$

$$S_{SAMPLE}(v) + S_{R\text{-}SAMPLE}(v) = 2. \qquad (5.b)$$

Furthermore, suppose that the example v is classified by the expert as either positive or negative. Then, one and only one of the following situations is true:

a) If (5.a) holds and v is a positive example, then system S_{SAMPLE} is not valid.
b) If (5.a) holds and v is a negative example, then system $S_{R\text{-}SAMPLE}$ is not valid.
c) If (5.b) holds and v is a positive example, then system $S_{R\text{-}SAMPLE}$ is not valid.
d) If (5.b) holds and v is a negative example, then system S_{SAMPLE} is not valid.

Therefore, the overall strategy, starting with two Boolean functions, is to attempt to generate a sequence of new examples $v_{k+1}, v_{k+2}, v_{k+3}, \ldots, v_m$, where each example is appropriately classified, as positive or negative, by the expert. Each additional example should have the property that it invalidates either S_{SAMPLE} or $S_{R\text{-}SAMPLE}$, i.e. one of the two Boolean functions must be modified. In doing so, it is expected that S_{SAMPLE} and $S_{R\text{-}SAMPLE}$ become more closely aligned with S_{HIDDEN} and $\bar{S}_{HIDDEN}$ respectively.

How does one find an example that invalidates either S_{SAMPLE} or $S_{R\text{-}SAMPLE}$? Conceptually it is quite simple. One strategy is to formulate and solve at most two *clause satisfiability problems*. The clause satisfiability problem has been examined with considerable success [Hooker, 1988a and 1988b]. A recent development reported in [Kamath *et al.*, 1992] uses an interior point algorithm developed by Karmakar and his associates [Karmakar, Resende, and Ramakrishnan, 1991] with considerable success. Some problem preprocessing techniques can be found in [Cavalier, Pardalos, and Soyster, 1990]). The interested reader may want to consult with the recent developments reported in the Second *DIAMACS* Challenge on Cliques, Coloring and Satisfiability [Trick and Johnson, 1995].

The two satisfiability problems for this problem are as follows: Determine an example $\bar{v}$ which results in a truth value TRUE for

$$\bar{S}_{SAMPLE}(\bar{v}) \quad \wedge \quad \bar{S}_{R\text{-}SAMPLE}(\bar{v}), \qquad \text{or:} \qquad (6.a)$$

$$S_{SAMPLE}(\bar{v}) \quad \wedge \quad S_{R\text{-}SAMPLE}(\bar{v}) \qquad (6.b)$$

If (6.a) is TRUE (i.e., satisfied), then $\bar{v}$ is evaluated as positive by both systems (Boolean functions), and if (6.b) is TRUE, $\bar{v}$ is evaluated as negative by both systems. Observe that relations (6.a) and (6.b) are equivalent to relations (5.a) and (5.b), respectively. If an example is found which satisfies either (6.a) or (6.b), then one of the two functions is modified and the same process is repeated. Next, suppose that no example can be found that satisfies (6.a) or (6.b). Does this mean that $S_{SAMPLE} \equiv S_{HIDDEN}$? Unfortunately, the answer is no. In this case we revert to a random search process.

A number of computer experiments was conducted in [Triantaphyllou and Soyster, 1996b] in order to investigate the effectiveness of the proposed strategy compared with random input learning (that is, when new examples are generated randomly). Those results are very encouraging because they suggest that a *"hidden system"* can be inferred, on the average, with at most 50% of the data required with random input. This guided learning approach was also applied on the Wisconsin breast cancer data base [Murphy and Aha, 1994] with remarkable success.

A recent development on inferring a small set of clauses from positive and negative examples is reported in [Deshpande and Triantaphyllou, 1996]. In that approach the main branch-and-bound algorithm of [Triantaphyllou, 1994] is combined with a very fast and simple heuristic. That heuristic uses a randomization approach for performing a local search while forming a single clause at time (according to the OCAT philosophy). Computational experiments suggest that the new development is even more promising. Some random problems with 7,000 and 18,000 examples which were defined on 15 Boolean attributes were solved in matter of a few hours on an IBM

3090-600S computer. Before that work, the largest problem solved was up to 1,000 examples defined on 32 atoms.

The above idea of using randomization in a search algorithm has been explored recently by other researchers as well. For instance, Feo and Resende in [1995] have successfully used randomization (the GRASP approach) to solve clause satisfiability (SAT) problems. Also, in a recent book Motwani and Raghavan [1995] provide a comprehensive presentation of the theory on randomized algorithms. Randomization also offers a natural and intuitive way for implementing parallelism in algorithms.

5 INFERENCE OF MONOTONE BOOLEAN FUNCTIONS

The previous sections dealt with the case of inferring general Boolean functions (either in CNF or DNF form). However, in many real life applications the behavior of a system is *monotone* [Boros, Hammer and Hooker, 1994]. In a continuous environment the previous statement means that the output of the system is consistent with the magnitude of the data.

In [Boros, Hammer and Hooker, 1994] the problem of inferring monotone Boolean functions is presented. In that treatment it is assumed that some observations are available which describe the behavior of a monotone Boolean function. Each observation is described by the values of some of the attributes (denoted by the vector x), while an unknown number of other attributes (denoted as the vector y) has undetermined values. The Boolean function may return one of a discrete set of values, often different values for the inputs which correspond to the same x part (but to different y parts). Next, some penalty coefficients are defined (in a geometric sense) and the goal then is to determine a function which minimizes the sum of all these penalties and also to be monotone.

Also note that in the previous treatment the problem of inquiring values of the function for certain new inputs was not considered. Neither was considered the problem of inferring a small size Boolean function. These are, however, the focus of the developments discussed in the following sections. In the next paragraphs we give the formal definitions for monotonicity and some key concepts.

Let $\alpha,\beta \in E_n$, E_n where denotes the set of all binary vectors of length n. Then, we say that the vector $\alpha = (\alpha_1,\alpha_2,\alpha_3,\ldots,\alpha_n)$ *precedes* the vector $\beta = (\beta_1,\beta_2,\beta_3,\ldots,\beta_n)$ (denoted as: $\alpha \leq \beta$) if and only if the following is true: $\alpha_i \leq \beta_i$, for all $1 \leq i \leq n$. A Boolean function $f(x)$ is *monotone* if for any vectors $\alpha,\beta \in E_n$, the relation $f(\alpha) \leq f(\beta)$ follows from the fact that $\alpha \leq \beta$. Let M_n be the set of all monotone Boolean functions defined on n variables. A binary vector α of length n is said to be the *upper zero* of a function $f(\alpha) \epsilon M_n$, if $f(\alpha) = 0$ and, for any vector $\beta \prec \alpha$, we have $f(\beta) = 0$.

A binary vector α of length n is said to be the *lower unit* of a function $f(\alpha) \epsilon M_n$, if $f(\alpha) = 1$ and, for any vector β from E_n such that $\beta \prec \alpha$, we get $f(\beta) = 1$. We will also call the above defined monotone Boolean function to be an *increasing (isotone) monotone* Boolean function in contrast with a decreasing monotone Boolean function. A Boolean function is *decreasing (antitone) monotone*, if for any vectors $\alpha, \beta \epsilon E_n$, the relation $f(\alpha) \leq f(\beta)$ follows from the fact that $\alpha \geq \beta$ [Rudeanu, 1974].

In terms of monotone Boolean function terminology note that the set of all upper zeros represents the *border elements* of the negative pattern. In an analogous manner we can also define the border of a positive pattern defined by using the lower units of the function. In this manner any monotone Boolean function represents two *"compact patterns"*. Please note that inferring the border elements of the positive and negative patterns is equivalent to the problem of inferring the structure of the target monotone Boolean function f from collections of positive and negative examples.

Some illustrative examples of monotone Boolean functions are: the constants 0 and 1, the identity function $f(x) = x$, the disjunction $x_1 \vee x_2$, the conjunction $x_1 \wedge x_2$, etc. Any function obtained by a composition of monotone Boolean functions is also monotone. In other words, the class of all monotone Boolean functions is closed. Moreover, the class of all monotone Boolean functions is one of the five *maximal (pre-complete) classes* in the set of all Boolean functions. That is, there is no closed class of Boolean functions, containing all monotone Boolean functions and distinct from the class of monotone functions and the class of all Boolean functions. The reduced *disjunctive normal form* (DNF) of any monotone Boolean function, distinct of 0 and 1, does not contain negations of variables. The set of functions $\{0, 1, (x_1 \vee x_2), (x_1 \wedge x_2)\}$ is a *complete system* (and moreover, a *basis*) in the class of all monotone Boolean functions [Alekseev, 1988].

Let $\psi(n)$ be the number of monotone Boolean functions defined on n variables. Then, 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, [Kleitman, 1969], and [Alekseev, 1988]).

Let a monotone Boolean function $f \epsilon M_n$ be defined with the help of a certain operator A_f (also called an *oracle*). That is, when f is fed with a vector $\alpha = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n)$, the operator A_f returns the value of $f(\alpha)$. The problem posed here is that of finding all upper and lower zeros of an arbitrary function $f \epsilon M_n$ with the help of a certain number of accesses to the operator A_f.

In the following paragraphs we state a very important theoretical result regarding the inference of monotone Boolean functions. Let $\mathcal{F} = \{F\}$ be the set of all algorithms which can solve the above monotone Boolean function inference problem and $\varphi(F, f)$ be the number of accesses to the operator A_f required to completely infer a monotone function $f \in M_n$.

Next, the Shannon function $\varphi(n)$ is introduced as follows [Hansel, 1966]:

$$\varphi(n) = \min_{F \in \mathcal{F}} \max_{f \in M_n} \varphi(F, f). \tag{7}$$

It was shown in [Hansel, 1966] that when inferring a monotone Boolean function the following relation is true (also known as *Hansel's lemma*):

$$\varphi(n) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}, \tag{8}$$

where $\lfloor n/2 \rfloor$ is the *floor* value of $n/2$.

It was shown in [Hansel, 1966] that restoration algorithms for monotone Boolean functions, and which use Hansel's lemma, are optimal in terms of the Shannon function. That is, they minimize the maximum input size requirements of any possible restoration algorithm. Finally, we state another related theorem [Kovalerchuk *et al.*, 1995]:

Theorem 6: *Each general Boolean function can be described in terms of several monotone Boolean functions.*

6 SOME KEY PROBLEMS AND ALGORITHM OUTLINES

FOR MONOTONE FUNCTIONS

PROBLEM 1: *The Main Inference Problem*
The central problem is how to infer a monotone Boolean function by issuing a sequence of membership inquires to an operator or "oracle". The solution of this problem is presented in detail in [Kovalerchuk *et al.*, 1996]. That approach uses the concept of Hansel chains and tries to identify border points of the previous "positive" and "negative" patterns. These patterns, in turn, lead to the definition of the target monotone Boolean function in DNF form. That algorithm is optimal in the sense of conditions (7) and (8), as explained in the previous section. There could be two different scenarios with this problem: one with no initial examples, and another with some initial examples.

PROBLEM 2: *Inference When Examples Have Different Costs*

The problem is similar to the fundamental problem presented in the previous paragraph. However, now it is assumed that different examples may be associated with different costs when one seeks to classify them by issuing a membership inquiry to an "oracle". In a related context, the user may somehow know the probability that a given example will have true (or, equivalently, false) value.

The basic idea of the proposed algorithm is as follows. Suppose that the example α, which needs to be classified by the "oracle" as part of the function restoration process, is associated with a known cost of classification, denoted as C_α. Moreover, suppose that the user knows (or can easily determine) the classification values of two other examples, say α_* and α^*, with costs $C_{\alpha*}$, and $C_\alpha*$, respectively which satisfy the following three conditions:

(*i*) $\alpha_* \leq \alpha \leq \alpha^*$ (i.e., α is between them),

(*ii*) $C_\alpha > (C_{\alpha*} + C_\alpha*)$. That is, their combined costs are lower than the cost of the target example α,

and (*iii*) $f(\alpha_*) = f(\alpha^*)$.

In a case like the above, the value of $f(\alpha)$ can be inferred from the values of two other examples at a lower cost than directly inquiring about the cost of the original target example α.

7. CONCLUDING REMARKS

In this paper we presented a survey of some results in inferring a small set of CNF or DNF clauses from positive and negative examples. A logical analysis approach was proposed for this purpose. The advantage of using a logical analysis approach is that the end result can be easily translated into a set of logical rules which can be more understandable by domain experts.

The paper considered the case of dealing with general Boolean functions and also the special case of dealing with monotone Boolean functions. For the case of general Boolean functions, the paper discussed a number of heuristics for inferring a small set of clauses and also some theoretical results on lower bounds on the number of clauses and ways for problem decomposition. This was achieved by exploiting the properties of a special graph which can be easily constructed from the input data.

For the monotone Boolean function case the main issue is that when inference is based in Hansel chains, then the target function can be inferred with a small number of membership inquires. Moreover, algorithms which use Hansel chains are optimal (in the sense of the Shannon function).

ACKNOWLEDGEMENTS

The first two authors gratefully recognize the partial support from the Office of Naval Research (ONR) grant N00014-95-1-0639 and LaSPACE (a NASA program). We thank the anonymous referee for his/her many thoughtful comments and suggestions made on an earlier version of this paper.

REFERENCES

ANGLUIN, D., 1988, "Queries and Concept learning," *Machine Learning*, Vol. 2, pp. 319-342.

ANGLUIN, D., 1992, "Computational Learning Theory: Survey and Selected Bibliography," *Proceedings of the 24-th Annual ACM Symposium on the Theory of Computing*, Victoria, BC, Canada, May 4-6, pp. 351-369.

ALEKSEEV, V.B., 1988, "Monotone Boolean Functions". Encyclopedia of Mathematics, v. 6, Kluwer Academic Publishers, 306-307.

BOROS, E., V. GURVICH, P.L. HAMMER, T. IBARAKI, AND A. KOGAN, 1994, "Structural Analysis and Decomposition of Partially Defined Boolean Functions," RRR 13-94, 25 pages, RUTCOR, Rutgers University, NJ.

BOROS, E., P.L. HAMMER, AND J.N. HOOKER, 1994, "Predicting Cause-Effect Relationships From Incomplete Discrete Observations," *SIAM J. on Discrete Math.*, Vol. 7, No. 4, pp. 531-543.

BLAIR, C.E., R.G. JEROSLOW, AND J.K. LOWE, 1985, "Some Results and Experiments in Programming Techniques for Propositional Logic," *Computers and Operations Research*. No. 13, pp.633-645.

CARRAGHAN, R., AND P.M. PARDALOS, 1990, "An Exact Algorithm for the Maximum Clique Problem," Operations Research Letters, 9, November, pp. 375-382.

CAVALIER, T.M., P.M. PARDALOS, AND A.L. SOYSTER, 1990, "Modeling and Integer Programming Techniques Applied to Propositional Calculus," J.P. Ignizio (ed.). *Computers and Operations Research*. 17:6, 561-570.

DESHPANDE, A.S., and E. TRIANTAPHYLLOU, 1996, "A Randomized Polynomial Heuristic for Inferring A Small Number of Logical Clauses from Examples," *Mathematical and Computer Modelling*, in print.

FEO, T.A. AND M.G.C. RESENDE, 1995, "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimization*, Vol. 6, pp. 109-133.

FU, L.M., 1993, "Knowledge-based connectionism for revising domain theories," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, No. 1, pp. 173-182.

HAMMER, P.L., AND E. BOROS, 1994, "Logical Analysis: An Overview," RUTCOR Research Report, Rutgers University, NJ.

HAMMER, P.L., AND A. KOGAN, 1992, "Horn Function Minimization and Knowledge Compression in Production Rule Bases," RUTCOR Research Report # 8-92, Rutgers University, NJ.

HANSEL, G., 1966, "Sur le nombre des fonctions Boolenes monotones den variables". *C.R. Acad. Sci. Paris*, v. 262, n. 20, 1088-1090.

HAUSSLER, D. 1989, "Learning conjunctive concepts in structural domains," *Machine Learning*, Vol. 4, pp. 7-40.

HAUSSLER, D., AND M. WARMUTH, 1993, "The Probably Approximately Correct (PAC) and Other Learning Models," Chapter in: *Foundations of Knowledge Acquisition: Machine Learning*, A.L. Meyrowitz and S. Chipman (Eds), Kluwer Academic Publishers, Norwell, MA, pp. 291-312.

HOOKER, J.N., 1988a, "Generalized Resolution and Cutting Planes," R.G. Jeroslow (ed.), *Annals of Operations Research*, Vol. 12, No. 1-4, pp. 217-239.

HOOKER, J.N., 1988b, "A Quantitative Approach to Logical Inference," *Decision Support Systems*. North-Holland, No. 4, pp. 45-69.

GIMPEL, J., 1965, "A Method of Producing a Boolean Function Having an Arbitrarily Prescribed Prime Implicant Table," *IEEE Trans. on Computers*, Vol. 14, pp. 485-488.

GOLDMAN, S.A., 1990, "Learning Binary Relations, Total Orders, and Real-Once Formulas." Ph.D. Thesis, Massachusetts Institute of Technology, September 1990. Available as Technical Report MIT/LCS/TR-483, MIT Laboratory for Computer Science.

GORBUNOV, Y. and B. KOVALERSCHUK, 1982, "An Interactive Algorithm For Restoring of a Monotone Boolean Function," *Izvestia AN USSR* (Proceedings of the Acad. of Sci.), STN, 2, Tashkent, pp. 3-16, in Russian.

JEROSLOW, R.G., 1988, "Computation-Oriented Reductions of Predicate to Prepositional Logic," *Decision Support Systems*. North-Holland, No. 4, pp. 183-197.

JEROSLOW, R.G., 1989, *Logic-Based Decision Support*, North-Holland.

QUINLAN, J.R., "Discovering rules by induction from large collections of examples," in *Expert Systems in the Micro Electronic Age*, D. Michie, Ed., Edinburgh, Scotland: Edinburgh University Press, 1979.

KAMATH, A.P., N.K. KARMAKAR, K.G. RAMAKRISHNAN, AND M.G.C. RESENDE, 1990, "Computational Experience with an Interior Point Algorithm on the Satisfiability Problem," *Annals of Operations Research*. P.M. Pardalos and J.B. Rosen (eds.). Special issue on: Computational Methods in Global Optimization, Vol 25, pp. 43-58.

KAMATH, A.P., N.K. KARMAKAR, K.G. RAMAKRISHNAN, AND M.G.C. RESENDE, 1992, "A Continuous Approach to Inductive Inference," *Math. Progr.*, Vol. 57, pp. 215-238.

KAMATH, A.P., N.K. KARMAKAR, K.G. RAMAKRISHNAN, AND M.G.C. RESENDE, 1994, "An Interior Point Approach to Boolean Vector Synthesis," *Proceedings of the 36th MSCAS*, pp. 1-5.

KARMAKAR, N.K., M.G.C. RESENDE, AND K.G. RAMAKRISHNAN, 1991, "An Interior Point Algorithm to Solve Computationally Difficult Set Covering Problems," *Math. Progr.*, Vol. 52, pp. 597-618.

KLEITMAN, D., 1969, "On Dedekind's problem: the number of monotone Boolean functions". *Proc. Amer. Math. Soc.* 21, 677-682.

KLEITMAN, D., 1969, "On Dedekind's problem: the number of monotone Boolean functions". Proc. Amer. Math. Soc. 21, 677-682.

KEARNS, M., MING LI, L. PITT, AND L.G. VALIANT, 1987, "On the Learnability of Boolean Formulae," *Journal of the Association for Computing Machinery*, No. 9, pp. 285-295.

KOVALERCHUK, B, E. TRIANTAPHYLLOU, and E. VITYAEV, 1995, "Monotone Boolean Function Learning Techniques Integrated with User Interaction," *Proceedings of the 12-th Inter'l Conference in Machine Learning*, Lake Tahoe, CA, U.S.A., July 9-12, pp. 41-48.

KOVALERCHUK, B., E. TRIANTAPHYLLOU, A.S. DESHPANDE, AND E. VITYAEV, 1996, "Interactive Learning of Monotone Boolean Functions," *Information Sciences*, 20 pages, (in print).

McCLUSKEY, E., 1956, "Minimization of Boolean Functions," *Bell Syst. Tech. J.*, Vol. 35, pp. 1417-1444.

MANGASARIAN, O.L., R. SETIONO, AND W.H. WOLBERG, 1991, "Pattern Recognition Via Linear Programming: Theory and Application to Medical Diagnosis," in: *Large-Scale Numerical Optimization*, Eds. T.F. Coleman, and Y. Li, SIAM, pp. 22-30.

MANSOUR, Y., 1992, "Learning of DNF Formulas", *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 53-59.

MOTWANI, R. AND P. RAGHAVAN, 1995, *Randomized Algorithms*, Cambridge University Press, New York, NY, USA.

MURPHY AND D.W. AHA, 1994, UCI Repository of Machine Learning Databases, Machine readable data repository, Irvine, CA, University of California, Dept. of Information and Computer Science.

PARDALOS, P.M., AND C.S. RENTALA, 1990, "Computational Aspects of a Parallel Algorithm to Find the Connected Components of a Graph," Technical Report, Dept. of Computer Science, Pennsylvania State University.

PEYSAKH, J., 1987, "A Fast Algorithm to Convert Boolean Expressions into CNF," IBM Computer Science RC 12913 (#57971), Watson, NY.

PITT, L. AND L.G. VALIANT, 1988, "Computational Limitations on Learning from Examples," *Journal of the Association for Computing Machinery*. Vol. 35, No. 4, pp. 965-984.

RUDEANU, S., 1974, *Boolean Functions and Equations*, North-Holland, NY.

SHAVLIK, J.W., 1994, "Combining Symbolic and Neural Learning," *Machine Learning*, Vol. 14, pp. 321-331.

TRIANTAPHYLLOU, E, A.L. SOYSTER, AND S.R.T. KUMARA, 1994, "Generating Logical Expressions from Positive and Negative Examples via a Branch-And-Bound Approach," *Computers and Operations Research*, Vol. 21, No. 2, pp. 185-197.

TRIANTAPHYLLOU, E., 1994, "Inference of A Minimum Size Boolean Function From Examples by Using A New Efficient Branch-And-Bound Approach," *Journal of Global Optimization*, Vol. 5, No. 1, pp. 69-94.

TRIANTAPHYLLOU, E. AND A.L. SOYSTER, 1995, "An Important Relationship Between CNF and DNF Systems Which are Derived from Examples," *ORSA Journal on Computing*, Vol. 7, No. 3, pp. 283-285.

TRIANTAPHYLLOU, E. AND A.L. SOYSTER, 1996a, "On the Minimum Number of Logical Clauses Which Can be Inferred From Positive and Negative Examples," *Computers and Operations Research*, Vol. 23, No. 8, pp. 783-799.

TRIANTAPHYLLOU, E. AND A.L. SOYSTER, 1996b, "An Approach to Guided Learning of Boolean Functions," *Computers and Mathematical Modelling*, Vol. 23, No. 3, pp. 69-86.

TRICK, M. AND D. JOHNSON, 1995, "Second DIAMACS Challenge on Cliques, Coloring and Satisfiability," American Mathematical Society, Rutgers University, Summer.

TURKSEN, I.B., AND H. ZHAO, "An Equivalence Between Inductive Learning and Pseudo-Boolean Logic Simplification: A Rule Generation and Reduction Scheme," *IEEE Transactions on Systems*, Man and Cybernetics, Vol. 23, No. 3, pp. 907-917, 1993.

VALIANT, L.G., 1984, "A Theory of the Learnable," *Comm. of ACM*, Vol. 27, No. 11, pp. 1134-1142.

WILLIAMS, H.P., 1986, "Linear and Integer Programming Applied to Artificial Intelligence," Preprint series, University of Southampton, Faculty of Mathematical Studies, pp. 1-33.

YABLONSKII, S., 1986, *Introduction to Discrete Mathematics*, Moscow, Nauka Publ. (in Russian).

CONTENTS

Ohseok Kwon
University of Maryland
College Park, Maryland, USA

Huan Li
NORTEL
Richardson, Texas, USA

Cormac Lucas
Brunel University
Uxbridge, Middlesex, UK

Gautam Mitra
Brunel University
Uxbridge, Middlesex, UK

Riad A. Mohammad
SABRE Decision Technologies
Dallas-Ft. Worth Airport, Texas, USA

Hossein Mousavi
Brunel University
Uxbridge, Middlesex, UK

B. Neveu
INRIO-CERMICS
Sophia-Antipolis Cedex, FRANCE

Soren S. Nielsen
University of Texas at Austin
Austin, Texas, USA

Rema Padman
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Jean-Yves Potvin
Université de Montréal
Montréal, Quebec, CANADA

L. C. Puryear
Software Solutions
Research Triangle Park, North Carolina, USA

Stephen F. Roehrig
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Thomas F. Siems
Federal Reserve Bank of Dallas
Dallas, Texas, USA

David M. Steiger
University of North Carolina at Greensboro
Greensboro, North Carolina, USA

Theodore B. Trafalis
University of Oklahoma
Norman, Oklahoma, USA

Evangelos Triantaphyllou
Louisana State University
Baton Rouge, Louisana, USA

M. Vidyasagar
Centre for AI and Robotics
Bangalore, INDIA

V. Vinay
Indian Institute of Science
Bangalore, INDIA

Edward A. Wasil
American University
Washington, DC, USA

Dan Zhu
University of Iowa
Iowa City, Iowa, USA

Yixin Zhu
NORTEL
Richardson, Texas, USA

Arkadiusz Zimniak
Poznań University of Technology
Poznań, POLAND

CONTRIBUTORS

Richard S. Barr
Southern Methodist University
Dallas, Texas, USA

M. Bouzoubaa
INRIA-CERMICS
Sophia-Antipolis Cedex, FRANCE

V. Chandru
Indian Instutute of Science
Bangalore, INDIA

Hai D. Chu
SABRE Decision Technologies
Dallas-Ft. Worth, Texas, USA

Nicolas P. Couellan
University of Oklahoma
Norman, Oklahoma, USA

Steven P. Coy
University of Maryland
College Park, Maryland, USA

Cihan H. Dagli
University of Missouri–Rolla
Rolla, Missouri, USA

Aniruddha Deshpande
Louisana State University
Baton Rouge, Louisana, USA

Eric Gelman
SABRE Decision Technologies
Dallas-Ft. Worth, Texas, USA

Fred Glover
University of Colorado
Boulder, Colorado, USA

Bruce L. Golden
University of Maryland
College Park, Maryland, USA

François Guertin
Université de Montréal
Montréal, Quebec, CANADA

G. Hasle
SINTEF
Oslo, NORWAY

Ellis L. Johnson
Georgia Institute of Technology
Atlanta, Georgia, USA

Joanna Józefowska
Poznań University of Technology
Poznań, POLAND

Jeffery L. Kennington
Southern Methodist University
Dallas, Texas, USA

Boris Kovalerchuk
Louisana State University
Baton Rouge, Louisana, USA

V. G. Kulkarni
University of North Carolina
Chapel Hill, North Carolina, USA

Distributors for North America:
Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, Massachusetts 02061 USA

Distributors for all other countries:
Kluwer Academic Publishers Group
Distribution Centre
Post Office Box 322
3300 AH Dordrecht, THE NETHERLANDS

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available from the Library of Congress.

INTERFACES IN COMPUTER SCIENCE AND OPERATIONS RESEARCH

Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies

EDITED BY

Richard S. BARR
Southern Methodist University
Dallas, Texas, USA

■

Richard V. HELGASON
Southern Methodist University
Dallas, Texas, USA

■

Jeffery L. KENNINGTON
Southern Methodist University
Dallas, Texas, USA

KLUWER ACADEMIC PUBLISHERS
Boston/London/Dordrecht