Chapter 2 [1]

THE ONE CLAUSE AT A TIME (OCAT) APPROACH TO DATA MINING AND KNOWLEDGE DISCOVERY

Evangelos Triantaphyllou[2]
Louisiana State University
Department of Computer Science
298 Coates Hall
Baton Rouge, LA 70803
U.S.A.
Email: trianta@lsu.edu
Web: *http://www.csc.lsu.edu/trianta*

Abstract: This chapter reviews a data mining and knowledge discovery approach called OCAT (for One Clause At a Time). The OCAT approach is based on concepts of mathematical logic and discrete optimization. As input it uses samples of the performance of the system (or phenomenon) under consideration and then it extracts its underlying behavior in terms of a compact and rather accurate set of classification rules. This chapter also provides ways for decomposing large scale data mining problems, and a way of how to generate the next best example to consider for training. The later methods can be combined with any Boolean function learning method and are not restricted to the OCAT approach only.

Key Words: Inductive Inference, Knowledge Discovery, Data Mining, Rule Extraction, Learning from Examples, CNF/DNF, Boolean Functions, Discrete Optimization, Maximum Clique, Connected Components in Graphs, Machine Learning.

---

[1] Triantaphyllou, E. and G. Felici (Eds.), **Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques**, Massive Computing Series, Springer, Heidelberg, Germany, pp. 45-87, 2005.

[2] The author is very appreciative for the support by the U.S. Navy, Office of Naval Research (ONR), research grants N00014-95-1-0639 and N00014-97-1-0632.

1. INTRODUCTION

In many scientific and engineering problems one often needs to study the operation of some system or phenomenon of interest. As input data we consider observations regarding the performance of this system under different situations. Moreover, we assume that these observations belong to disjoint classes. These observations may describe the different states of nature of the system under consideration. That is, each observation is comprised by a set of variables or attributes along with their corresponding values. Often there are only two types of states of nature or classes of observations. The first class may correspond to failures of the system and similarly, the second class may correspond to its successes. Then, a central problem is to utilize the above information (i.e., the observations grouped into different classes) and construct a set of classification rules (also known as decision rules or just rules) which can explain the behavior of the system.

Such situations may arise in many settings. For instance, the system of interest may be some kind of a mechanical device. Then observations are descriptions of the operation of this device. Such descriptions may provide information on the noise level, vibration level, temperature of different parts of the device, fluid levels, revolution speed of certain parts, etc. The different classes may be the properly functioning and malfunctioning states of this device. Then one may be interested in studying data that describe different conditions when the device is functioning properly and also when it is malfunctioning and extract any patterns that might be embedded in these data. Such patterns may be used later to accurately predict the state of this device when one has not determined yet if it is functioning properly or not, but has information on some key performance characteristics as described above.

As another example, one may wish to consider observations that describe certain clinical and non-clinical characteristics associated with studies to determine whether a given patient has a certain medical condition such as breast cancer. Observations may now describe family history facts, the presence or not of lesions in mammographic images, blood analysis results, etc. The two disjoint classes, roughly speaking, may be the benign or malignant nature of any lesions present in a patient's breasts. Usually, such final diagnosis takes place after a specimen from the lesion is analyzed in a lab (i.e., by performing a biopsy). In such cases one may wish to identify any patterns in collections of such observations which may lead to an accurate diagnosis without having to perform an invasive and costly procedure such as a biopsy.

Similar situations, like the ones described above, may arise when one studies other medical conditions, the reliability of mechanical and electronic systems, weather phenomena, characterization of images into different classes (for digital image analysis), prediction of financial events (for example, for investing type of decision-making), prediction of buying behaviors of consumers, and so on.

Until recently such inference problems have been studied by statistical models, such as regression and factor analysis. However, the proliferation of effective and efficient computing power, along with the easy availability of magnetic storage media and new ways for gathering data fast and efficiently, created the need for developing new methods for analyzing large amounts of data. Moreover, data may now be highly heterogeneous and also highly unstructured. The later is the case more and more now with the advent and proliferation of the World Wide Web (WWW).

The above are the main reasons for the emergence of the new computational discipline called *data mining and knowledge discovery from databases (DM&KDD)* [Fayyad, *et al.*, 1996]. This new discipline is based on the use of many computational methods formally known as *artificial intelligence* (AI) classification methods. Currently, such classification methods include standard back-propagation neural networks, nearest neighbor methods, discriminant analysis, cluster analysis, and linear programming based methods. Such techniques attempt to generalize from collections of available classified data. Therefore, they rely on the supposition that the more representative the data are, the more accurate the performance of these methods is.

However, there are some basic weaknesses in using these techniques. For example, according to Johnson [1991] the use of Bayesian models may be controversial, if not unethical (for instance, in medical diagnosis), because the fundamental requirement of strict randomness rarely occurs in reality. Also, standard back-propagation neural networks techniques are problematic because they do not provide an explanation of their decision-making process [Fu, 1993]. In summary, there are two closely related fundamental weaknesses with most of the existing classification methods:

First weakness:

The way these classification methods work and produce recommendations may not be appealing to domain experts. For instance, the decision-making process inside a standard back-propagation neural network might be awkward to many users with no engineering or computer science background.

Second weakness:

The available training data are often insufficient to guaranty statistically significant results. This is especially true for data which describe applications in diagnostic systems. As result these methods may be unreliable for some of the real life applications or their reliability cannot be scientifically guaranteed.

The severity of the first weakness is increasingly recognized lately and some hybrid systems try to combine rule based systems with neural network techniques. Such hybrid systems produce a set of decision rules which in turn can be used for the classification of new instances. Newly developed intelligent hybrid systems, and in particular knowledge based neural networks [Fu, 1993, Galant, 1988; Bradshaw, *et al*, 1989; Hall and Romanyuk, 1990; Towell, *et al.*, 1990; and Sun and Alexandre, 1997], appear to be more promising. Unfortunately, these systems have the potential to create an exponential number of rules [Shavlik, 1994]. Moreover, they may even produce contradictory rules because they are not built in a complete logic-based framework.

The second weakness is usually treated by a brutal force approach. That is, by collecting huge amounts of training data. However, this may be a too time and cost consuming remedy. More importantly, when one considers the number of all possible states of nature, then millions, or even billions, of observations may represent only a tiny fraction of the entire state space. This may cause severe concerns regarding the reliability of the extracted knowledge, especially for medical diagnostic systems (see, for example, [Kovalerchuk, *et al.*, 2000]). For instance, a system described on 50 binary attributes corresponds to $2^{50} = 1.12589 \times 10^{15}$ different states of nature. In that case, even a few billions of observations may be considered as been statistically too few.

This chapter presents some developments for inferring a compact Boolean function from collections of positive and negative examples and also some related subjects. This chapter is organized as follows. The next section describes the main problems to be examined and some related developments from the literature. Section 3 defines the notation to be used throughout this chapter. Section 4 focuses on the problem of inferring a small number of classification rules from two collections of training (positive and negative) examples. These rules are first extracted in the form of a compact Boolean function. Some optimal algorithms and fast heuristics are described. It also describes a data transformation method for converting multi-valued data into binary ones. Section 5 describes a guided learning approach. Section 6 presents the notion of a special graph which can be easily built from the training examples. Section 7 describes how that graph can be used to decompose a large size data mining problem. A detailed illustrative example is presented in Section 8. Section 9 is the last one and describes some conclusions and possible future research topics.

2. SOME BACKGROUND INFORMATION

We assume that some observations are available and they describe the behavior of a system of interest. It is also assumed that the behavior of this system is fully described by a number, say n, of *attributes* (also known as *atoms*, *parameters*, *variables*, *characteristics*, *predicates*, or just *features*). Thus, each observation is defined by a vector of size n. The i-th (for $i = 1, 2, 3, ..., n$) element of such a vector corresponds to the value of the i-th attribute. These attributes may be of any data type. For instance, they may take continuous, discrete, or binary (i.e., 0/1) values. Each observation belongs to one and only one of K distinct classes. It is assumed that the observations are *noise free*. Furthermore, it is also assumed that the class membership associated with each observation is the correct one.

One may assume that some observations, say m, are already available. New observations (along with their class membership) may become available later but the analyst has no control on their composition. In addition to the previous scenario, the analyst may be able to define the composition of new observations (i.e., to set the values of the n attributes) and then perform a test, or ask an expert or *"oracle"*, to determine the class membership of a new observation. The main goal is to use the available classified observations to extract the underlying behavior of the target system in terms of a pattern. Next, this pattern is used to, hopefully, accurately infer the class membership of unclassified observations.

The extraction of new knowledge in the form of a set of logical decision rules from collections of classified data is a particular type of learning from examples. The related literature is vast and is increasing rapidly and thus it will not be exhaustively discussed. One of the most recent contributions is the book by Truemper [2004] which discusses methods for inferring the logic of a system of interest from sampled observations and then use it towards building intelligent systems. 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* (for Probably Approximately Correct) learning theory (see, for instance, [Angluin, 1988] and [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 [Valiant, 1985; Rivest, 1987; and 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]. 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.

Another issue is the sample complexity of a learning algorithm. That is, the number of examples needed to accurately approximate a target concept. The presence of bias in the selection of a hypothesis from the hypothesis space can be beneficial in reducing the sample complexity of a learning algorithm. Usually the amount of bias in the hypothesis space H is measured in terms of the *Vapnik-Chernovenkis dimension*, denoted as *VCdim(H)* [Haussler, 1988].

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. This can be motivated for achieving the maximum possible simplicity (Occam's razor) and easy validation of the derived new knowledge.

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]). Some early related work in this area is due to [Bongard, 1970]. The classical approach to deal with this Boolean function inference problem as a minimization problem (in the sense of minimizing the number of CNF or DNF clauses) was developed in [Quine, 1952 and 1955] and [McCluskey, 1956]. However, the exact versions of the Quine-McCluskey algorithm cannot handle large scale problems. Thus, some heuristic approaches have been proposed. These heuristics include the systems MINI [Hong, *et al.*, 1974], PRESTO [Brown, 1981], and ESPRESSO-MV [Brayton, *et al.*, 1985]. Another widely known approach in dealing with this problem is the use of *Karnaugh maps* [Karnaugh, 1953]. However, this approach cannot be used to solve large scale problems [Pappas, 1994]. Another application of Boolean function minimization can be found in the domain of multicast [Chang, *et al.*, 1999] where one needs a minimum number of keys.

A related method, denoted as SAT (for satisfiability), has been proposed in [Kamath, *et al.,* 1992]. In that approach one first pre-assumes an upper limit on the number of clauses to be considered, say k. Then a clause satisfiability (SAT) model is formed and is solved by using an interior point method developed by Karmakar and his associates [Karmakar, Resende, and Ramakrishnan, 1992]. If the 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 yields a system with the minimum number of clauses. It is important to observe at this point that from the computational point of view it is much harder to prove that a given SAT problem is infeasible than to prove that it is feasible. Therefore, trying to determine a minimum size Boolean function by using the SAT approach may be computationally too difficult. Some computational results indicate that the branch-and-bound (B&B) approach proposed in [Triantaphyllou, 1994] is significantly more efficient than the previous satisfiability based approach (5,500 times faster on the average for those tests).

In [Felici and Truemper, 2002] the authors propose a different use of the SAT model. They formulate the problem of finding a clause with maximal coverage as a minimum cost satisfiability (MINSAT) problem and solve such problem iteratively by using the logic SAT solver *Leibniz*, which was developed by Truemper [Truemper, 1998]. That method is proved to be computationally feasible and effective in practice. The same authors also propose several variants and extensions to that system, some of which are discussed in Chapter 6 of this book. Further extensions on this learning approach are also discussed in [Truemper, 2004].

A closely related problem is to study the construction of a partially defined Boolean function (or pdBf), not necessarily of minimal size, given disjoint sets of positive and negative examples. That is, now it is required that the attributes of the function be grouped according to a given scheme (called a decomposition structure) [Boros, *et al.*, 1994]. Typically, a pdBf may have exponentially many different extensions.

In summary, the most recent advances in distinguishing between observations in two or more classes can be classified into six distinct categories. These developments are; the clause satisfiability approach to inductive inference by Kamath, *et al.* [1992; and 1994]; some B&B and heuristic approaches of generating a small set of logical decision rules developed in [Triantaphyllou, *et al.*, 1994], and [Triantaphyllou, 1994]; some improved polynomial time and NP-complete cases of Boolean function decomposition by [Boros, *et al.*, 1994]; some MINSAT formulations [Felici and Truemper, 2002]; decision tree based approaches [Quinlan, 1979; and 1986]; linear programming based approaches by [Wolberg and Mangasarian, 1990], [Mangasarian, *et al.*, 1990] and [Mangasarian, *et al.*, 1995]; some approaches which combine symbolic and connectionist machines (neural networks) as proposed by [Sun and Alexandre, 1997], Shavlik [1994], Fu [1993], Goldman and Sloan [1994] and Cohn, *et al.* [1994] and finally, some nearest neighbor classification approaches by Hattori and Torri [1993], Kurita [1991], Kamgar-Parsi and Kanal [1985].

The main challenge in inferring a target set of discriminant decision rules from positive and negative examples is that the user can *never* be absolutely certain about the correctness of the decision rules, unless he/she has processed the entire set of all possible examples which is of size 2^n in the binary case. In the general case this number is far higher. Apparently, even for a small value of n, this task may be practically impossible to realize.

Fortunately, many real life applications are governed by the behavior of a *monotone* system or they can be described by *a combination of a small number of monotone systems*. In data mining and knowledge discovery research monotonicity offers some unique computational advantages. By knowing the value of certain examples, one can easily infer the values of more examples. This, in turn, can significantly expedite the learning process. This chapter discusses the case of inferring general Boolean functions from disjoint collections of training examples. The case of inferring a monotone Boolean function is discussed in Chapter 4 of this book as written by Torvik and Triantaphyllou [2005].

3. DEFINITIONS AND TERMINOLOGY

Let $\{A_1, A_2, A_3, ..., A_t\}$ be a set of t Boolean *attributes*. Each attribute A_i (i = 1, 2, 3, ..., t) can be either true (denoted by 1) or false (denoted by 0). Let F be a *Boolean function* over these attributes. For instance, the expression $(A_1 \vee A_2) \wedge (A_3 \vee \bar{A}_4)$ is such a Boolean function, where "$\vee$" and "$\wedge$" stand for the logical *"OR"* and *"AND"* operators, respectively. That is, F is a mapping from $\{0,1\}^t \rightarrow \{0,1\}$ which determines for each combination of truth values of the attributes $A_1, A_2, A_3, ..., A_t$ 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 $F(v) = 1$. Similarly, the *negative examples* are the vectors $v \in \{0,1,\}^t$ such that $F(v) = 0$. Therefore, given a function F defined on the t attributes $\{A_1, A_2, A_3, ..., A_t\}$, then a vector $v \in \{0,1,\}^t$ is either a positive or a negative example. Equivalently, we say that a vector $v \in \{0,1,\}^t$ is *accepted* (or *rejected*) by a Boolean function F if and only if the vector v is a positive (or a negative) example of F. For instance, let F be the Boolean function $(A_1 \vee A_2) \wedge (A_3 \vee \bar{A}_4)$. Consider the two vectors $v_1 = (1,0,0,0)$ and $v_2 = (1,0,0,1)$. Then, it can be easily verified that $F(v_1) = 1$. That is, the vector v_1 is a positive example of the function F. However, the vector v_2 is a negative example of F (since $F(v_2) = 0$).

At this point some additional definitions are also introduced. Let $e \in \{0,1,\}^t$ be an example (either positive or negative). Then, $\hat{e} \in \{0,1,\}^t$ is defined as the *complement of the example e*. For instance, if e = (0,1,1,0,0,0), then $\hat{e}$ = (1,0,0,1,1,1). Similarly, let E be a collection of examples. Then, $\hat{E}$ is defined as the *complement of the collection E*. A Boolean expression is in CNF or DNF if it is in the form (*i*) or (*ii*), respectively:

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

and

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

where a_i is either A_i or $\bar{A}_i$ and ρ_j is the set of indexes.

In other words, a CNF expression is a conjunction of disjunctions, while a DNF expression is a disjunction of conjunctions.

It is known [Peysakh, 1987] that any Boolean function can be transformed into the CNF or DNF form. 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.

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 $\hat{E}^-$ (considered as the positive examples) and $\hat{E}^+$ (considered as the negative examples).

This theorem is stated here because the graph theoretic developments throughout this chapter assume that a system is derived in CNF form. However, since a clause inference algorithm which derives DNF expressions (such as, for instance, the SAT approach described in [Kamath, *et al.,* 1992; and 1994]) can also derive CNF expressions (by applying the previous theorem), the methods in this chapter are applicable both to CNF and DNF cases.

In summary, a set of positive examples is denoted as E^+ and a set of negative examples is denoted as E^-. Given these two sets of positive and negative examples, the constraints to be satisfied by a system (i.e., a Boolean function) are as follows. In the CNF case, each positive example should be accepted by all the disjunctions in the CNF expression and each negative example should be rejected by at least one of the disjunctions. In the case of DNF systems, any positive example should be accepted by at least one of the conjunctions in the DNF expression, while each negative example should be rejected by all the conjunctions.

4. THE ONE CLAUSE AT A TIME (OCAT) APPROACH

The main ideas are best described via a simple illustrative example. Suppose that the data in Table 1 represent some sampled observations of the function of a system of interest. Each observation is described by the value of two *continuous* attributes denoted as A_1 and A_2. Furthermore, each observation belongs to one of two classes, denoted as Class 1 and Class 2. A number of problems can be considered at this point. The main problem is how to derive a pattern, in the form of a set of rules, that is consistent with these observations. As set of rules we consider here logical clauses in the CNF (conjunctive normal form) or DNF (disjunctive normal form). That is, we seek the extraction of a Boolean function in CNF or DNF form.

Although, in general, many such Boolean functions can be derived, the focus of the proposed approach is the derivation of a function of *minimum size*. By minimal size we mean a Boolean function which consists of the minimum number of CNF or DNF clauses. We leave it up to the analyst to decide whether he/she wishes to derive CNF or DNF functions. The proposed methodology can handle both cases when Theorem 1, as described in the previous section, is used.

Table 1. Continuous Observations for Illustrative Example.

Example No.	A_1	A_2	Class No.	Example No.	A_1	A_2	Class No.
1	0.25	1.50	1	12	1.00	0.75	1
2	0.75	1.50	1	13	1.50	0.75	1
3	1.00	1.50	1	14	1.75	0.75	2
4	0.50	1.25	1	15	0.50	0.50	1
5	1.25	1.25	2	16	1.25	0.50	2
6	0.75	1.00	1	17	2.25	0.50	2
7	1.25	1.00	1	18	2.75	0.50	2
8	1.50	1.00	2	19	1.25	0.25	2
9	1.75	1.00	1	20	1.75	0.25	2
10	2.25	1.00	2	21	2.25	0.25	2
11	0.25	0.75	1				

4.1 Data Binarization

Next it is demonstrated how the continuous data depicted in Table 1 can be represented by equivalent observations with only binary attributes.

This is achieved as follows. First we start with the first continuous attribute, i.e., attribute A_1 in this case, and we proceed until we cover all the attributes. It can be observed from Table 1 that the *ordered* set, denoted as $Val(A_1)$, with all the values of attribute A_1 is defined as the following ordered list:

$$\begin{aligned} Val(A_1) &= \{V_i(A_1), \text{ for } i = 1, 2, 3, ..., 9\} = \\ &= \{0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.25, 2.75\}. \end{aligned}$$

Obviously, the cardinality of this set is less than or at most equal to the number of all available observations. In this instance, the cardinality is equal to 9. Next, we introduce 9 binary attributes $A_{1,i}^{/}$ (for $i = 1, 2, 3, ..., 9$) as follows:

$$A_{1,i}^{/} = \begin{cases} 1, & \textit{iff} \quad A_1 \leq V_i(A_1), \ \textit{for} \quad i = 1,2,3,...,9, \\ 0, & \textit{otherwise.} \end{cases}$$

In general, the previous formula becomes for any multi-valued attribute A_j:

$$A_{j,i}^{/} = \begin{cases} 1, & \text{iff} \quad \mathrm{A_j} \leq \mathrm{V_i(A_j)}, \ \text{for} \quad \mathrm{i} = 1, 2, 3, ..., \mathrm{M}, \\ 0, & \textit{otherwise.} \end{cases}$$

For instance, by using the above introduced binary attributes, from the second observation (i.e., vector (0.75, 1.50) we get:

$$\{A_{1,1}^{/}, A_{1,2}^{/}, A_{1,3}^{/}, A_{1,4}^{/}, A_{1,5}^{/}, A_{1,6}^{/}, A_{1,7}^{/}, A_{1,8}^{/}, A_{1,9}^{/}\} = \{1, 1, 1, 0, 0, 0, 0, 0, 0\}.$$

Similarly, for the second continuous attribute A_2 the set $Val(A_2)$ is defined as follows:

$$\begin{aligned} Val(A_2) &= \{V_i(A_2), \text{ for } i = 1, 2, 3, ..., 6\} = \\ &= \{0.25, 0.50, 0.75, 1.00, 1.25, 1.50\}. \end{aligned}$$

Working as above, for the second observation we have:

$$\{A_{2,1}^{/}, A_{2,2}^{/}, A_{2,3}^{/}, A_{2,4}^{/}, A_{2,5}^{/}, A_{2,6}^{/}\} = \{1, 1, 1, 1, 1, 1\}.$$

The above transformations are repeated for each one of the non-binary attributes. In this way, the transformed observations are defined on *at most* mxn binary attributes (where m is the number of observations and n is the original number of attributes). The precise number of the transformed attributes can be easily computed by using the following formula:

$$\sum_{i=1}^{n} |Val(A_i)|,$$

where $|s|$ denotes the cardinality of set s.

The binary attributed observations which correspond to the original data (as presented in Table 1) are presented in Table 2 (parts (*a*) and (*b*)).

Table 2 (*a*). The Binary Representation of the Observations in the Illustrative Example (first set of attributes for each example).

Example No.	**First set of attributes:** $A_{1,i}^{/}$, for $i = 1, 2, 3, \ldots, 9$.								
	$A_{1,1}^{/}$	$A_{1,2}^{/}$	$A_{1,3}^{/}$	$A_{1,4}^{/}$	$A_{1,5}^{/}$	$A_{1,6}^{/}$	$A_{1,7}^{/}$	$A_{1,8}^{/}$	$A_{1,9}^{/}$
1	1	0	0	0	0	0	0	0	0
2	1	1	1	0	0	0	0	0	0
3	1	1	1	1	0	0	0	0	0
4	1	1	0	0	0	0	0	0	0
5	1	1	1	1	1	0	0	0	0
6	1	1	1	0	0	0	0	0	0
7	1	1	1	1	1	0	0	0	0
8	1	1	1	1	1	1	0	0	0
9	1	1	1	1	1	1	1	0	0
10	1	1	1	1	1	1	1	1	0
11	1	0	0	0	0	0	0	0	0
12	1	1	1	1	0	0	0	0	0
13	1	1	1	1	1	1	0	0	0
14	1	1	1	1	1	1	1	0	0
15	1	1	0	0	0	0	0	0	0
16	1	1	1	1	1	0	0	0	0
17	1	1	1	1	1	1	1	1	0
18	1	1	1	1	1	1	1	1	1
19	1	1	1	1	1	0	0	0	0
20	1	1	1	1	1	1	1	0	0
21	1	1	1	1	1	1	1	1	0

Table 2 (*b*). The Binary Representation of the Observations in the Illustrative Example (second set of attributes for each example).

Example	**Second set of attributes:** $A_{2,i}^{/}$, for $i = 1, 2, 3, \ldots, 6$.						Class
No.	$A_{2,1}^{/}$	$A_{2,2}^{/}$	$A_{2,3}^{/}$	$A_{2,4}^{/}$	$A_{2,5}^{/}$	$A_{2,6}^{/}$	No.
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1
4	1	1	1	1	1	0	1
5	1	1	1	1	1	0	2
6	1	1	1	1	0	0	1
7	1	1	1	1	0	0	1
8	1	1	1	1	0	0	2
9	1	1	1	1	0	0	1
10	1	1	1	1	0	0	2
11	1	1	1	0	0	0	1
12	1	1	1	0	0	0	1
13	1	1	1	0	0	0	1
14	1	1	1	0	0	0	2
15	1	1	0	0	0	0	1
16	1	1	0	0	0	0	2
17	1	1	0	0	0	0	2
18	1	1	0	0	0	0	2
19	1	0	0	0	0	0	2
20	1	0	0	0	0	0	2
21	1	0	0	0	0	0	2

From the way the binary attributes have been defined, it follows that the two sets of observations are equivalent to each other. However, the observations in Table 1 are defined in continuous attributes while the observations in Table 2 are defined in binary ones.

Given the above considerations, it follows that the original problem has been transformed into the binary problem depicted in Table 2 (parts (*a*) and (*b*)). This problem has the following two sets of positive and negative examples, denoted as E^{+} and E^{-}, respectively.

$$E^{+}=\begin{bmatrix}
1&0&0&0&0&0&0&0&0&1&1&1&1&1&1\\
1&1&1&0&0&0&0&0&0&1&1&1&1&1&1\\
1&1&1&1&0&0&0&0&0&1&1&1&1&1&1\\
1&1&0&0&0&0&0&0&0&1&1&1&1&1&0\\
1&1&1&0&0&0&0&0&0&1&1&1&1&0&0\\
1&1&1&1&1&0&0&0&0&1&1&1&1&0&0\\
1&1&1&1&1&1&1&0&0&1&1&1&1&0&0\\
1&0&0&0&0&0&0&0&0&1&1&1&0&0&0\\
1&1&1&1&0&0&0&0&0&1&1&1&0&0&0\\
1&1&1&1&1&1&0&0&0&1&1&1&0&0&0\\
1&1&0&0&0&0&0&0&0&1&1&0&0&0&0
\end{bmatrix}, \text{ and}$$

$$E^{-}=\begin{bmatrix}
1&1&1&1&1&0&0&0&0&1&1&1&1&1&0\\
1&1&1&1&1&1&0&0&0&1&1&1&1&0&0\\
1&1&1&1&1&1&1&1&0&1&1&1&1&0&0\\
1&1&1&1&1&1&1&0&0&1&1&1&0&0&0\\
1&1&1&1&1&0&0&0&0&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&0&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&0&0&0&0\\
1&1&1&1&1&0&0&0&0&1&0&0&0&0&0\\
1&1&1&1&1&1&1&0&0&1&0&0&0&0&0\\
1&1&1&1&1&1&1&1&0&1&0&0&0&0&0
\end{bmatrix}.$$

Finally, it should be stated here that Chapter 7 of this book [Bartnikowski, *et al.*, 2005] presents a detailed study of the binarization problem.

4.2 The One Clause At a Time (OCAT) Concept

As it was mentioned in the previous section, the problem of deriving a Boolean function from sets of observations has been extensively studied in the literature. In our setting each example was a binary vector of size *n* (number of binary attributes). The proposed method employs an approach which constructs one clause at a time, called the OCAT (for One Clause At a Time) approach. That approach is greedy in nature in the sense that the first clause (in the CNF case) accepts all the positive examples while it rejects as many negative examples as possible. The second clause also accepts all positive examples, but rejects as many negative examples from the ones not rejected so far. Consecutive clauses are generated in a similar manner until all the derived clauses reject the entire set of negative examples. The operation of the OCAT approach is best described in Figure 1. In this figure E^{+} represents the set with the positive examples while E^{-} is the set with the negative ones.

The core of the OCAT approach is step 2, in Figure 1. In Triantaphyllou, *et al.* [1994] a branch-and-bound (B&B) based algorithm is presented which solves the problem posed in step 2. A more efficient B&B algorithm, along with other enhancements, are described in Triantaphyllou [1994]. The OCAT approach returns the set of desired clauses (i.e., the CNF system) as set C.

```
$i = 0$ ; $C = \emptyset$ ; {initializations}
**DO WHILE** $(E^- \neq \emptyset)$
        **Step 1:** $i \leftarrow i + 1$ ;
        **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 $C \leftarrow C \wedge c_i$ ;
        **Step 5:** Let $E^- \leftarrow E^- - E^-(c_i)$ ;
**REPEAT;**
```

Figure 1. The One Clause At a Time (OCAT) Approach (for the CNF case).

4.3 A Branch-and-Bound Approach for Inferring Clauses

This B&B algorithm is best described in [Triantaphyllou, 1994]. The basic steps are described next in terms of an illustrative example. Consider the following two sets of positive and negative examples:

$$E^+ = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \text{ and } E^- = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

These examples are defined on four attributes (and their negations). Recall that for the CNF case, the requirement is that the clause to accept all the positive examples, while rejecting as many negative examples as possible. Next, define as $POS(A_i)$ the set of the positive examples which are accepted by a CNF clause when the attribute A_i is included in that clause. For instance, for the previous examples, one has (please note that for simplicity only the indexes of these examples are used): $POS(A_2) = \{1,2\}$, $POS(\bar{A}_3) = \{1,2,4\}$, etc. That B&B algorithm also uses the concept of the $NEG(A_i)$ set which is defined in a similar manner.

The *search states* are described in terms of *two sets*. The first set refers to the *positive examples* which are accepted by the attributes which correspond to the arcs which connect that state (node) with the root node.

Similarly, the second set refers to the *negative examples* which are accepted by the attributes which correspond to the arcs which connect that state with the root node. Suppose that we are at state $S_i = [P_i, N_i]$ (where P_i, N_i correspond to the previous two sets of positive and negative examples, respectively). Now assume that the search considers the state (node) which is derived by following the arch which corresponds to attribute A_k. Then, the new state is: $S_j = [P_j, N_j]$, where the new sets P_j and N_j are defined as follows:

$$P_j = P_i \cup POS(A_k), \text{ and}$$
$$N_j = N_i \cup NEG(A_k).$$

Therefore, the search continues until terminal states are reached. A state $S_i = [P_i, N_i]$ is a *terminal state* if and only if: $P_i = E^+$ (i.e., it refers to *all positive* examples). Apparently, a terminal state with a *minimum cardinality* of the set N_i is *optimal* (in the OCAT sense). In the light of the previous considerations, the problem (for the CNF case) to be solved by the B&B search can be summarized as follows (where a_i is either A_i or $\bar{A}_i$):

Find a set of attributes S such that the following two conditions are true:

$$\left| \bigcup_{a_i \in S} NEG(a_i) \right| = minimum,$$

and

$$\left| \bigcup_{a_i \in S} POS(a_i) \right| = E^+.$$

The attributes in the S set are the ones that correspond to the attributes of the CNF clause to be constructed. Given the above definitions some useful derivations are possible. We say that a state S_i *absorbs* another state S_j if by expanding the state S_j, we cannot reach any better terminal state than the ones derived by expanding the state S_i. In such a case we call that the state S_j is an *absorbed state*. From the previous considerations it becomes obvious that once a state can be identified to be an absorbed state, then it can be dropped from further consideration. Then the following two theorems [Triantaphyllou, 1994] are applicable (only) when a CNF clause is to be generated and they provide some conditions for identifying absorbed states.

Theorem 2:
*The state $S_i = [P_i, N_i]$ **absorbs** the state $S_j = [P_j, N_j]$ if the following condition is true:* $P_j \subseteq P_i$ *and* $N_i \subseteq N_j$.

Theorem 3:
*Suppose that $S_i = [P_i, N_i]$ is a **terminal** state. Then, any state $S_j = [P_j, N_j]$, such that $|N_j| \geq |N_i|$, is **absorbed** by the state S_i.*

From the previous considerations it follows that there is a great advantage to reach terminal nodes early in the search process. In this way, the minimum size of their N_i sets can be used to effectively fathom search states. For these reasons that B&B search can be applied in *two phases*. During the first phase only a very small number (say, 10) of active states is maintained. If there are more than 10 active states, then they are ranked according to their P_i and N_i sizes. In this way, the states with the highest potential of being optimal are kept into memory. This is the principle of *beam search* in artificial intelligence (see, for instance, [Dietterich and Michalski, 1983]). At the end of phase one, a terminal state of small cardinality becomes available. Next, phase two is initiated. During the second phase a larger number (say, 50) of active states is allowed. However, states now can be fathomed more frequently because the size of a small N_i set of a terminal state is known.

An important issue with the previous two phases is to be able to decide when a terminal state is optimal (in the OCAT sense). As it was mentioned above, memory limitations may force the search to drop states which are not absorbed by any other state. Therefore, *there is a possibility to drop a state which could had lead to an optimal state (and thus to determine an optimal clause).*

Suppose that L non-absorbed states had to be dropped because of memory limitations. Let K_1, K_2, K_3, ..., K_L represent the cardinalities of their corresponding N_i sets. Next, define the quantity K_{MIN} as the minimum of the previous L numbers. Similarly, suppose that the B&B process has identified N terminal states. Let Y_1, Y_2, Y_3, ..., Y_N represent the cardinalities of their corresponding N_i sets. Define as Y_{MIN} the minimum of the previous N cardinalities. Then, the previous considerations lead to the proof of the following theorem [Triantaphyllou, 1994] which states a condition for establishing optimality.

Theorem 4:

*A terminal state $S_i = [P_i, N_i]$ is also an **optimal state** if the following two conditions are true:*

$$|N_i| = Y_{MIN}, \quad \text{and} \quad K_{MIN} \geq Y_{MIN}.$$

Note that this theorem can be applied after each one of the two phases. Obviously, if it is applicable after the first phase, then the second phase does not need to be initiated. The following lemma states a condition when optimality is not provable.

Lemma 1:

If $K_{MIN} < Y_{MIN}$, then an optimal clause accepts no less than K_{MIN} negative *examples.*

This lemma indicates that if optimality cannot be proven, then it is still possible to establish a *lower limit* on the number of negative examples which can be accepted by an optimal clause (or, equivalently, an upper limit on the number of negative examples which can be *rejected* by an optimal clause).

4.4 Inference of the Clauses for the Illustrative Example

When the OCAT algorithm, with the B&B approach described in [Triantaphyllou, 1994] is used in step 2, two Boolean functions can be derived. The first function is derived when the examples in set E^+ are used as the positive examples while the examples in E^- are used as the negative examples. We call the set of these clauses the *positive rules* (because the positive examples evaluate these clauses as true).

The Boolean function derived from the previous E^+ and E^- examples has the following form (note that the attribute names have been slightly altered to reflect the adjusted notation):

$$(\bar{A}_{1,8}' \wedge A_{2,2}' \wedge \bar{A}_{1,5}') \vee (A_{2,3}' \wedge \bar{A}_{1,8}' \wedge \bar{A}_{2,5}' \wedge \bar{A}_{1,6}') \vee$$
$$(A_{2,3}' \wedge A_{1,6}' \wedge \bar{A}_{1,8}' \wedge A_{1,7}' \wedge A_{2,4}') \vee (\bar{A}_{1,7}' \wedge A_{1,6}' \wedge \bar{A}_{2,4}').$$

Similarly, the second function is derived when the examples in set E^- are used as the positive examples while the examples in E^+ are used as the negative examples. Thus, we call these clauses the *negative rules*. The Boolean function derived from the previous E^- and E^+ examples is:

$$(A_{1,5}' \wedge \bar{A}_{2,3}') \vee (A_{1,5}' \wedge A_{1,6}' \wedge A_{1,7}' \wedge A_{1,8}') \vee$$
$$(A_{1,5}' \wedge A_{2,5}') \vee (A_{1,6}' \wedge A_{1,7}' \wedge \bar{A}_{2,4}') \vee (A_{1,6}' \wedge \bar{A}_{1,7}' \wedge A_{2,4}').$$

When the definitions of the Boolean attributes $A_{1,i}'$ (for i = 1, 2, 3, ..., 9) and $A_{2,j}'$ (for j = 1, 2, 3, ..., 6) are used, then it is easy to verify that the previous two functions yield the following two sets of rules defined on the two original continuous attributes A_1 and A_2:

(*i*) Positive Classification Rules:

($A_1 \leq 1.00$ and $A_2 \geq 0.5$)	(Rule R^+_1)
($A_1 \leq 1.25$ and $1.00 \geq A_2 \geq 0.75$)	(Rule R^+_2)
($A_1 = 1.75$ and $A_2 \geq 1.00$)	(Rule R^+_3)
($A_1 = 1.50$ and $0.75 \geq A_2$)	(Rule R^+_4)

(*ii*) Negative Classification Rules:

($A_1 \geq 1.25$ and $0.50 \geq A_2$)	(Rule R^-_1)
($A_1 \geq 2.25$)	(Rule R^-_2)
($A_1 \geq 1.25$ and $A_2 \geq 1.25$)	(Rule R^-_3)
($A_1 \geq 1.75$ and $0.75 \geq A_2$)	(Rule R^-_4)

($A_1 = 1.50$ and $A_2 \geq 1.00$) (Rule R^-_5)

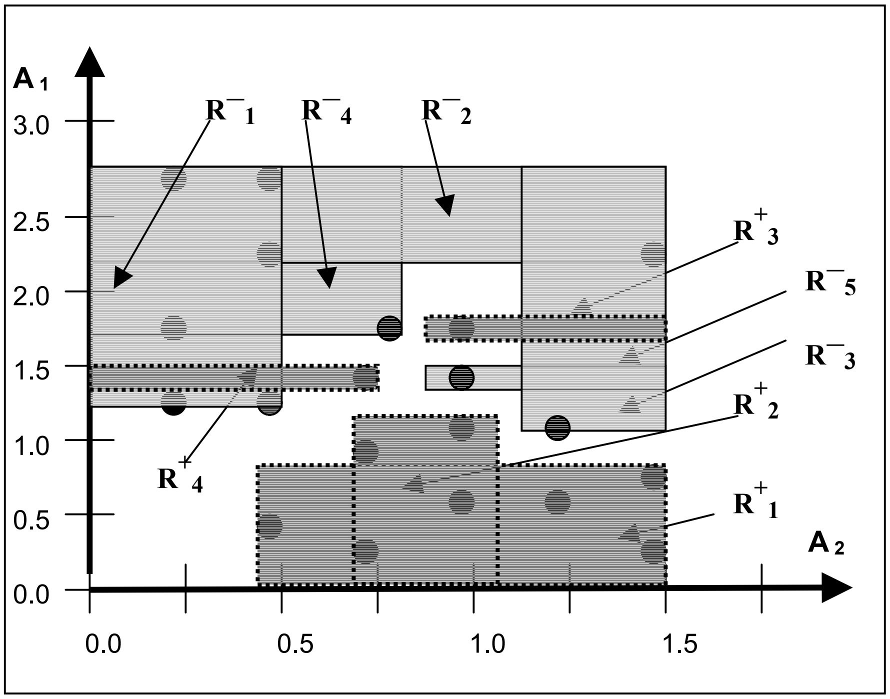

Figure 2. Continuous Data for Illustrative Example and Extracted Sets of Classification Rules.

When dealing with DNF clauses (as is the previous case), decision rules can be derived by observing that for an example to be positive, it must satisfy at least one clause. Thus, for the first of the previous clauses (i.e., for clause ($\bar{A}_{1,8}^{/} \wedge A_{2,2}^{/} \wedge \bar{A}_{1,5}^{/}$) and by noting that: $\bar{A}_8^{/} \leftarrow \bar{A}_{1,8}^{/}$, $A_{11} \leftarrow A_{2,2}^{/}$, and $\bar{A}_5^{/} \leftarrow \bar{A}_{1,5}^{/}$) the corresponding decision rule is:

IF *($\bar{A}_8$ and A_{11} and $\bar{A}_5$ are all true),*
THEN *(this example is a positive one).*

For the CNF case, a conjunction can be transformed into an equivalent logical decision rule by observing that the following two expressions are equivalent:

($A_1 \vee A_2$) is equivalent to: ($\bar{A}_1 \rightarrow A_2$).

From the above discussion it follows that any CNF or DNF expression with *k* clauses (conjunctions or disjunctions) can be described in terms of the same number *k* of decision rules. In the CNF case the attribute(s) which compose the "**IF**" parts of these rules are not uniquely determined. Therefore, it is the task of the field expert to decide which attributes are allowed to be present in the "**IF**" part of a rule (or *antecedent* part) or in the "**THEN**" part of a rule (or *consequent* part). It can be easily shown that given two classes of observations, then one can derive as many CNF conjunctions as the number of negative examples [Triantaphyllou, Soyster, and Kumara, 1994; or Triantaphyllou, 1994] (similar results hold for the DNF case). Therefore, for cases in which the negative examples are numerous one may be interested in determining a minimal (or at least very small) number of such logical decision rules. This is an issue of significant practical importance, since compact sets of decision rules are easier to validate and use.

The previous two sets of rules, along with the positive and negative examples (as defined in terms of the two continuous attributes A_1 and A_2) are depicted in Figure 2. The same figure also indicates some of the reasons why the proposed approach, at least for the binary case, delivered more accurate results when it was compared in [Deshpande and Triantaphyllou, 1998] with some other approaches (neural networks and separating planes via the LP approach developed by Mangasarian and his associates [Mangasarian, *et al.*, 1991]). When both sets of decision rules are used, then for a new observation to be classified as positive, it must be both accepted by the positive rules and also rejected by the negative rules (analogously for an observation to be classified as negative).

However, many existing classification techniques consider only one set of rules. Therefore, in the proposed approach there are three different classification decisions: *"Positive," "Negative,"* and *"Undecided."* Many traditional approaches do not consider the third type (i.e., *"Undecided"*). By forcing the derived sets of decision rules to be as compact as possible, the proposed approach has a tendency to isolate and *"close-in"* observations into compact groups defined in the same class. Many methods simply try to determine separating planes (borders) of some sort which are in the middle of some type of distance. Usually, such a distance reflects how apart the two classes of observations are. In this chapter the population space is actually partitioned into four types of areas: *"positive areas," "negative areas," "areas of conflict"* between the two sets of rules, and *"areas not covered"* by any rules. In Figure 2 the case *"areas of conflict"* between the two sets of rules does not occur (by coincidence).

To offset the drawback of the exponential time complexity of the B&B algorithm in step 2 of the OCAT approach, in [Deshpande and Triantaphyllou, 1998] a heuristic of polynomial time complexity is proposed. Under that heuristic the clause which is formed during a single iteration rejects *many* (as opposed to *as many as possible*) negative examples. This heuristic seems to offer an alternative approach when the problem size is very large. It can also be combined with the previous B&B approach and is also randomized as a GRASP (Greedy Random Adaptive Search Procedure [Feo and Resende, 1995]) approach.

4.5 A Polynomial Time Heuristic for Inferring Clauses

To offset the drawback of the exponential time complexity of the B&B algorithm in step 2 of the OCAT approach, in this heuristic clauses are formed in a manner such that each clause accepts all the examples in the E^+ set while it attempts to reject *many* (as opposed to *as many as possible* in the B&B approach) examples in the E^- set. Note that this is the main procedural difference between the B&B algorithm and the proposed heuristics. In the proposed heuristic this is achieved by choosing the attributes to form a clause based on an evaluative function (to be described later). Only attributes with high values in terms of the evaluative function are included in the current clause. A single clause is completely derived when all the examples in the E^+ set are accepted. The clause forming procedure is repeated until all the examples in the E^- set are rejected by the proposed set of clauses. As some computational results in [Deshpande and Triantaphyllou, 1998] indicate, this strategy may often result in Boolean functions with a small number of clauses.

Observe that if always the attribute with the *highest value* of the evaluative function is included in the clause, then there is an inherent danger of being trapped in a local optimal point. To prevent this undesirable behavior, a *randomized* approach is used. In this randomized approach, instead of a single attribute being included in a clause due to its highest value of the evaluative function, a candidate list is formed of attributes whose values in terms of the evaluative function are close to the highest value as derived from the evaluative function. Next, an attribute is *randomly* chosen out of the candidate list and is included in the CNF clause being derived.

Please note that it is possible for a CNF clause to reject as many negative examples as possible (and, of course, to accept all positive examples) but the entire system not to have a small (ideally minimum) number of clauses. Recall that the proposed heuristics follow the OCAT approach (see also Figure 1). That is, sometimes it may be more beneficial to have a less *"effective"* clause which does not reject a large number of negative examples, and still derive a system with very few clauses. Such systems are possible to derive with the use of randomized algorithms. A randomized algorithm, with a sufficiently large number of random replications, is more difficult to be trapped by a local optimal point.

A heuristic approach, termed ***RA1*** (for *Randomized Algorithm 1*), was proposed in [Deshpande and Triantaphyllou, 1998] to solve the first research problem considered in this chapter. Before the RA1 heuristic is formally presented, some new definitions and terminology are summarized next.

Definitions:

C =	The set of attributes in the current clause (disjunction).
A_k =	An attribute such that $A_k \in A$, where A is the set of all attributes $A_1, \ldots, A_n$.
$POS(A_k)$ =	The number of all positive examples in E^+ which would be accepted if attribute A_k is included in the current clause.
$NEG(A_k)$ =	The number of all negative examples in E^- which would be accepted if attribute A_k is included in the current clause. Please note that the last two definitions are slightly different from the previous definitions of POS and NEG as sets of examples. Now we are interested in the sizes of these sets only.
l =	The size of the candidate list.
$ITRS$ =	The number of times the clause forming procedure is repeated.

As an illustrative example of the above definitions, consider the following sets of positive and negative examples (which were also given in Section 3.3).

$$E^+ = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \text{ and } E^- = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

The set A of all attributes for the above set of examples is:

$$A = \{A_1, A_2, A_3, A_4, \bar{A}_1, \bar{A}_2, \bar{A}_3, \bar{A}_4\}.$$

Therefore, the $POS(A_k)$ and the $NEG(A_k)$ values are:

$POS(A_1) = 2$	$NEG(A_1) = 4$	$POS(\bar{A}_1) = 2$	$NEG(\bar{A}_1) = 2$
$POS(A_2) = 2$	$NEG(A_2) = 2$	$POS(\bar{A}_2) = 2$	$NEG(\bar{A}_2) = 4$
$POS(A_3) = 1$	$NEG(A_3) = 3$	$POS(\bar{A}_3) = 3$	$NEG(\bar{A}_3) = 3$
$POS(A_4) = 2$	$NEG(A_4) = 2$	$POS(\bar{A}_4) = 2$	$NEG(\bar{A}_4) = 4$

```
DO for ITRS number of iterations
BEGIN;
    DO WHILE ($E^- \neq \emptyset$)
      $C = \emptyset$; {initialization}
        DO WHILE ($E^+ \neq \emptyset$)
            Step 1: Rank in descending order all attributes $a_i \in a$ (where $a_i$ is either $A_i$ or $\bar{A}_i$) according to their $POS(a_i)/NEG(a_i)$ value. If $NEG(a_i) = 0$, then $POS(a_i)/NEG(a_i) = 1{,}000$ (i.e., an arbitrarily high value);
            Step 2: Form a candidate list of the attributes which have the $l$ top highest $POS(a_i)/NEG(a_i)$ values;
            Step 3: Randomly choose an attribute $a_k$ from the candidate list;
            Step 4: Let the set of attributes in the current clause be $C \leftarrow C \vee a_k$;
            Step 5: Let $E^+(a_k)$ be the set of members of $E^+$ accepted when $a_k$ is included in the current CNF clause;
            Step 6: Let $E+ \leftarrow E^+ - E^+(a_k)$;
            Step 7: Let $a \leftarrow a - a_k$;
            Step 8: Calculate the new $POS(a_i)$ values for all $a_i \in a$;
        REPEAT

      Step 9:  Let $E^-(C)$ be the set of members of $E^-$ which are rejected by $C$;
      Step 10: Let $E^- \leftarrow E^- - E^-(C)$;
      Step 11: Reset $E^+$;
    REPEAT
END;

CHOOSE the final Boolean system among the previous ITRS systems which has the smallest number of clauses.
```

Figure 3. The RA1 Heuristic [Deshpande and Triantaphyllou, 1998].

The problem now is to derive a small set of logical clauses which would correctly classify all the above examples. Suppose that there exists a *"hidden"* system given by the following Boolean function:

$$(A_2 \vee A_4) \wedge (\bar{A}_2 \vee \bar{A}_3) \wedge (A_1 \vee A_3 \vee \bar{A}_4).$$

It can be easily seen that the above Boolean function correctly classifies all the previous examples. Therefore, the first problem is to accurately estimate the above *"hidden"* system. This is accomplished by using heuristic RA1, as described in Figure 3.

The following theorem [Deshpande and Triantaphyllou, 1998] states an upper bound on the number of clauses which can be inferred by RA1 (where m_2 is the number of negative examples).

Theorem 5:
The RA1 approach terminates within at most m_2 iterations.

Next, let n be the number of attributes in the data set, m_1 be the number of examples in the E^+ set and m_2 be the number of examples in the E^- set. Then Theorem 6 [Deshpande and Triantaphyllou, 1998] states the time complexity of the RA1 algorithm.

Theorem 6:
The RA1 algorithm has a polynomial time complexity of order $O(n(m_1+m_2)m_1\, m_2\, ITRS)$.

From the way the $POS(A_k)$ and $NEG(A_k)$ values were defined, some critical observations can be made. When an attribute with a rather high value of the *POS* function is included in the CNF clause being formed, then chances are that some additional positive examples will be accepted by that clause as result of the inclusion of that attribute. Similarly, attributes which correspond to low *NEG* values, are likely not to cause many new negative examples to be accepted as result of the inclusion of that attribute in the current clause. Therefore, it makes sense to include as attributes in the CNF clause under formation, the ones which correspond to high *POS* values and, at the same time, to low *NEG* values.

In this chapter the notations $POS(a_i)/NEG(a_i)$ and $POS(A_k)/NEG(A_k)$ will be used interchangeably to denote the same concept. For the current illustrative example, the values of the $POS(A_k)/NEG(A_k)$ ratios are:

$POS(A_1)/NEG(A_1) = 0.5$	$POS(\bar{A}_1)/NEG(\bar{A}_1) = 1.0$
$POS(A_2)/NEG(A_2) = 1.0$	$POS(\bar{A}_2)/NEG(\bar{A}_2) = 0.5$
$POS(A_3)/NEG(A_3) = 0.33$	$POS(\bar{A}_3)/NEG(\bar{A}_3) = 1.0$

$$POS(A_4)/NEG(A_4) = 1.0 \qquad POS(\bar{A}_4)/NEG(\bar{A}_4) = 0.5$$

The above discussion illustrates the motivation for considering as possible candidates for the evaluative function, the functions: *POS/NEG*, *POS-NEG*, or some type of a weighted version of the previous two expressions. Some exploratory computational experiments indicated that the evaluative function *POS/NEG* was the most effective one. That is, it led to the formation of Boolean functions with less clauses than when the other evaluative functions were considered.

The randomization of the RA1 algorithm is done as follows. In step 2, the first l attributes with the highest value of the $POS(A_k) / NEG(A_k)$ ratio are chosen as the members of the candidate list and an attribute in the list was randomly chosen out of the candidate list in step 3. This is done in order to obtain different solutions at each iteration and prevent the system from being trapped by a locally optimal point.

In choosing a fixed value for the size l of the candidate list, there is a possibility that an attribute with a very low value of $POS(A_k) / NEG(A_k)$ ratio could be selected if the value of l is large enough (how large depends on the current data). That could occur if there are not l attributes with a sufficiently high value of the $POS(A_k) / NEG(A_k)$ ratio. If an attribute with a low value of $POS(A_k) / NEG(A_k)$ is chosen to be included in the clause, then the clause would accept less examples from the E^+ set or accept more examples from the E^- set, or both. All these three situations should be avoided as it would lead to an increase in the number of attributes in a clause (if it accepts less examples from the E^+ set) or, to an increase in the number of clauses (if the attribute accepts more examples from the E^- set), or both. To prevent the above situation from happening, a candidate list is formed of attributes, each of whose $POS(A_k) / NEG(A_k)$ value is within a certain percentage, say *" %,* of the highest value of the $POS(A_k) / NEG(A_k)$ value in the current candidate list. This ensures that the attribute (randomly chosen out of the candidate list) to be included in the clause has a value close to the highest value of the $POS(A_k) / NEG(A_k)$ ratios.

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 to solve clause satisfiability (SAT) problems. Also, in a 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.

To obtain a system with a very small number of clauses, the whole procedure is subjected to a certain number of iterations (denoted by the value of the *ITRS* parameter) and the system which has the least number of disjunctions is chosen as the final inferred Boolean system.

Referring to the previous illustrative example, if $l = 3$, then the values of the 3 best $POS(A_k) / NEG(A_k)$ ratios are: {1.0, 1.0, 1.0} (note that it is a coincidence that the three values are identical) which correspond to the attributes $\bar{A}_1$, A_2 and A_4, respectively. Let attribute A_2 be the randomly selected attribute from the candidate list. Note that attribute A_2 accepts examples number 2 and 3 from the current E^+ set. Therefore, at least one more attribute is required to complete the formation of the current clause. The whole process of finding a new attribute (other than attribute A_2 which has already been selected) with a very high value of POS/NEG is repeated. Now, suppose that the attribute with a high POS/NEG value happened to be A_4. It can be observed now that, when attributes A_2 and A_4 are combined together, they accept all the elements in the E^+ set. Therefore, the first clause is $(A_2 \vee A_4)$.

This clause fails to reject examples number 2, 3 and 6 in the E^- set. Therefore, examples number 2, 3 and 6 in the original E^- set constitute the reduced (and thus new) E^- set. The above process is repeated until a set of clauses are formed which, when combined together, reject all the examples in the original E^- set. Therefore, a final Boolean function for this problem could be as follows (recall that the algorithm is a randomized one and thus it may not return the same solution):

$$(A_2 \vee A_4) \wedge (\bar{A}_2 \vee \bar{A}_3) \wedge (A_1 \vee A_3 \vee \bar{A}_4).$$

5. A GUIDED LEARNING APPROACH

The above partitioning of the population of all possible examples into the previous four disjoint regions (also recall Figure 2), suggests a natural way to select the next example to classify by the expert (*"oracle"*) when new examples are selected for training. If the new (and thus unclassified) example is selected from the region which represents *"areas of conflict,"* then when it is classified by the expert it will indicate that at least one of the positive or negative sets of rules needs to be changed (since it has to be either positive or negative). Similarly, when the new example is selected from the region which represents *"areas not covered,"* then again when it is classified by the expert it will indicate that at least one of the positive or negative sets of rules needs to be changed. This realization is in direct agreement with the guided learning approach recommended in [Triantaphyllou and Soyster, 1995b].

The above observations are better formalized as follows. Let us consider two sets of positive and negative examples, denoted as E^+ and E^-, respectively, defined on t (binary or multi-valued) attributes. Let S_{SAMPLE} denote the set of rules (systems) derived from the sample data, i.e. when the examples in E^+ are classified as positive and the examples in E^- are classified as negative. Similarly, define as $S_{R\text{-}SAMPLE}$ the set of rules (system) derived when E^{-} is used as the positive examples while E^{+} as the negative examples. That is S_{SAMPLE} is the set with the positive rules while $S_{R\text{-}SAMPLE}$ is the set with the negative rules. Also, define S_{HIDDEN} as the *"hidden logic"* system and $\hat{S}_{HIDDEN}$ (please note the "^" symbol on top of "*S*") as the complement of S_{HIDDEN}. The guided learning strategy proposed in [Triantaphyllou and Soyster, 1995b] is based on the following theorem (which is valid for the binary and also the multi-valued case):

Theorem 7:

Suppose that there exists an example v such that:

$$S_{SAMPLE}(v) + S_{R\text{-}SAMPLE}(v) = 0, \text{ or:} \quad (1)$$

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

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

a) If (1) holds and v is a positive example,
then system S_{SAMPLE} is not valid.

b) If (1) holds and v is a negative example,
then system $S_{R\text{-}SAMPLE}$ is not valid.

c) If (2) holds and v is a positive example,
then system $S_{R\text{-}SAMPLE}$ is not valid.

d) If (2) holds and v is a negative example,
then system S_{SAMPLE} is not valid.

Therefore, the overall strategy, starting with two sets of rules, 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. Each additional example should have the property that it invalidates either S_{SAMPLE} or $S_{R\text{-}SAMPLE}$, i.e. one of the two sets of rules 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 $\hat{S}_{HIDDEN}$, respectively. Finally, as it was shown in [Triantaphyllou and Soyster, 1995b], the next example can be determined by solving a clause satisfiability problem.

During this guided learning approach one may observe that the current Boolean functions need to be modified only when a new training example indicates that a Boolean function is inaccurate (by misclassifying it). In [Nieto Sanchez, *et al.*, 2002] some algorithms are proposed which modify a Boolean function in a way that the new function correctly classifies the new example (and also all the previous training examples) and does so by performing a minimal (kind of *"surgical"*) modification. That is, these algorithms select a clause of the current function and modify it. The algorithms in [Triantaphyllou and Soyster, 1995b] and [Nieto Sanchez, *et al.*, 2002] have the potential to expedite the guided learning process both in terms of the number of the new training examples needed to accurately infer a *"hidden"* logic but also in terms of the time required to update the inferred Boolean functions.

6. 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. To understand the motivation for introducing this graph, consider a situation with $t = 5$ attributes. 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*. For the positive example v_1, note that $A_1, \bar{A}_2, A_3, \bar{A}_4,$ and A_5 are true (or, equivalently, $\bar{A}_1, A_2, \bar{A}_3, A_4$ and $\bar{A}_5$ are false). Similar interpretations exist for the remaining two examples v_2 and v_3.

6.1 The Definition of the Rejectability Graph

Denote by *ATTRIBUTES(v)* the set of the attributes that are true (have value "1") for a particular (either positive or negative) example *v*. With this definition, one obtains from the above data:

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

Next consider a single CNF clause (i.e., a disjunction), denoted as *C*, of the general form:

$$C = \bigvee_{i=1}^{M} a_i \quad \text{(where } a_i \text{ is either } A_i \text{ or } \bar{A}_i\text{).}$$

The clause *C* *accepts* an example *v* (i.e., *v* is a positive example of *C*) if and only if at least one of the attributes in the set *ATTRIBUTES(v)* is also one of the attributes in the expression:

$$\bigvee_{i=1}^{M} a_i \, .$$

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.

Now observe that there is no *single CNF clause* which can *simultaneously* reject the two negative examples v_2 and v_3, while at the same time accept the positive example v_1. This is true because any clause which simultaneously rejects the two examples v_2 and v_3, should not contain any of the attributes in the *union* of the two sets given as *ATTRIBUTES(*v_2*)* and

ATTRIBUTES(v_3). But, if none of the attributes of the set $\{A_1, A_2, \bar{A}_2, A_3, A_4, \bar{A}_4, A_5\}$ = *ATTRIBUTES(v_2)* U *ATTRIBUTES(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 any clause which accepts the positive example v_1, the previous two negative examples v_2 and v_3 cannot also be *rejected* by such clause.

From the above considerations it follows that given three examples v_1, v_2, and v_3, then the examples v_2 and v_3 are rejectable by a single clause (disjunction), subject to the example v_1, if and only if the following condition is true:

ATTRIBUTES(v_1) é *ATTRIBUTES(v_2)* U *ATTRIBUTES(v_3)*.

In general, given a set of positive examples E^+, then two negative examples v_1 and v_2 are rejectable by a single clause if and only if the condition in the following theorem [Triantaphyllou and Soyster, 1996] is satisfied:

Theorem 8:

Let E^+ be a set of positive examples and v_1 , v_2 be two negative examples. There exists a CNF clause which accepts all the positive examples and rejects both negative examples v_1 and v_2 if and only if:

ATTRIBUTES(v_i) é *ATTRIBUTES(v_1)* U *ATTRIBUTES(v_2)*,

for each positive example $v_i \in E^+$.

The above theorem follows directly from the previous considerations. Given two collections of positive and negative examples, denoted as E^+ and E^-, respectively, Theorem 8 motivates the construction of a graph $G = (V, E)$ as follows:

$$V = \{ V_1, V_2, V_3, ..., V_{M2} \},$$

where M_2 is the cardinality of E^- (i.e., each vertex corresponds to one negative example in E^-), and

$$e \in E, \quad \text{where } e = (V_i, V_j),$$

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 denote this graph as the *rejectability graph* (or *the R-graph*) of E^+ and E^-. The previous theorem indicates that it is computationally straightforward to construct this graph. If there are M_2 negative examples, then the maximum number of edges is $M_2(M_2 — 1)/2$. Therefore, the rejectability graph can be constructed by performing $M_2(M_2 — 1)/2$ simple rejectability examinations.

An Illustrative Example

Consider the following E^+ and E^- sets (given earlier and repeated here):

$$E^{+} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \text{ and } E^{-} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Since there are 6 negative examples, there are 6(6 — 1)/2 = 15 possible pairwise comparisons (i.e., single rejectability tests). For instance, the first (v_1) and third (v_3) negative examples correspond to the vertices V_1 and V_3, respectively. Next one can observe that because:

$ATTRIBUTES(v_1) \cup ATTRIBUTES(v_3) = \{A_1, A_2, A_3, A_4, \bar{A}_2, \bar{A}_4\}$,

and $ATTRIBUTES(v_i)$ é $\{A_1, A_2, A_3, A_4, \bar{A}_2, \bar{A}_4\}$, for each $v_i \in E^{+}$,

it follows that there is an edge which connects the vertices V_1 and V_3 in the rejectability graph. The rejectability graph G, which corresponds to this illustrative example, is presented in Figure 4.

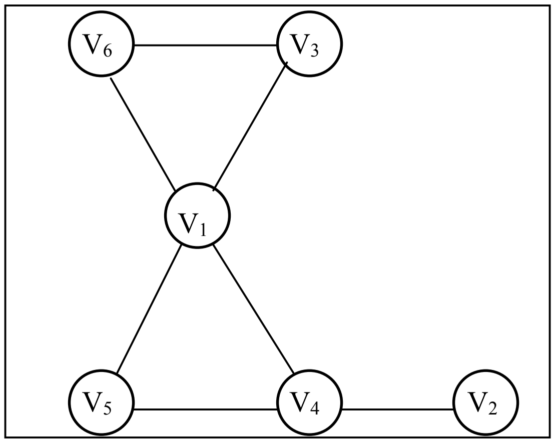

Figure 4. The Rejectability Graph for E^{+} and E^{-}.

6.2 Properties of the Rejectability Graph

The rejectability graph G of a set of positive and a set of negative examples possesses a number of interesting properties. Two of these properties refer to the cliques of the rejectability graph. A *clique* of a graph is a subgraph in which all the nodes are connected with each other. The *minimum clique cover number* (denoted as *k(G)*) is the smallest number of cliques needed to cover the vertices of G (see, for instance, [Golumbic, 1980] and [Bollobás, 1979]). The following theorem [Triantaphyllou and Soyster, 1996] refers to any clique of the rejectability graph.

Theorem 9:
Suppose that the two sets E^+ and E^- are given and β is a subset of k negative examples from E^- ($k \leq$ size of set E^-) with the property that the subset can be rejected by a single CNF clause which also accepts each one of the positive examples in E^+. Then, the vertices corresponding to the k negative examples in the rejectability graph G form a clique of size k.

The previous theorem states that any set of negative examples which can be rejected by a single clause corresponds to a clique in the rejectability graph. However, *the converse is not true*. That is, not every clique in the rejectability graph corresponds to a set of negative examples which can be rejected by a single clause. To see this consider the following illustrative example.

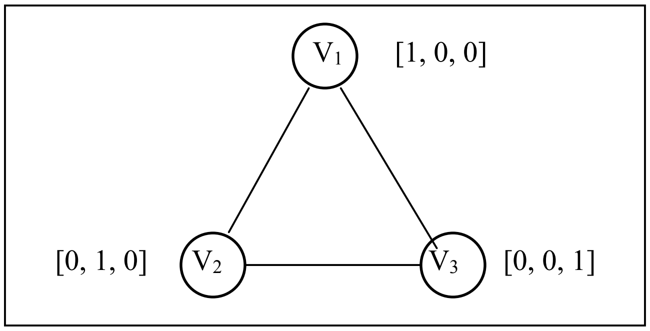

Figure 5. The Rejectability Graph for the Second Illustrative Example.

An Illustrative Example

Consider the following sets E^+ and E^-:

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

It can be easily verified that any pair of the three negative examples in E^- can be rejected by a single clause which also accepts the positive example in E^+. For instance, the first and second negative examples are rejected by the clause (A_3), which also accepts the positive example in E^+. Similarly, the first and third negative examples can be rejected by (A_2), while (A_1) rejects the second and third examples. In all cases, these clauses accept the single example in E^+. Therefore, the corresponding rejectability graph is a triangle (i.e., a clique with three nodes, see also Figure 5). However, a clause which would reject all the three negative examples should *not include* any attributes from the following set:

$$\begin{aligned} &ATTRIBUTES(v_1) \cup ATTRIBUTES(v_2) \cup ATTRIBUTES(v_3) = \\ &= ATTRIBUTES((1, 0, 0)) \cup ATTRIBUTES((0, 1, 0)) \cup \\ &\cup ATTRIBUTES((0, 0, 1)) = \\ &= \{A_1, A_2, A_3, \bar{A}_1, \bar{A}_2, \bar{A}_3\}. \end{aligned}$$

Obviously, no such clause exists when $n = 3$. Therefore, a minimum size set of CNF clauses which satisfy the requirements of the current examples is: $(A_3) \vee (A_2)$, which is of size 2.

6.3 On the Minimum Clique Cover of the Rejectability Graph

Consider two sets of positive and negative examples E^+ and E^-, respectively. Let $\hat{G}$ be the *complement* of the rejectability graph G of the two sets of examples. Recall that the complement of a graph is constructed as follows: The complement graph has exactly the same vertices as the original graph. There is an edge between any two vertices if and only if there is no edge between the corresponding vertices of the original graph. Next, define $\omega(\hat{G})$ as the *size of the maximum clique* of the graph $\hat{G}$ and $k(G)$ as the minimum clique cover number of the rejectability graph G. Let r be the minimum number of CNF clauses required to reject all the examples in E^-, while accepting all the examples in E^+. Then, the following theorem [Triantaphyllou and Soyster, 1996]] states a *lower bound* (i.e., the minimum clique cover $k(G)$) on the *minimum number* of clauses required to reject all the negative examples in E^-, while accepting all the positive examples in E^+.

Theorem 10:
Suppose that E^+ and E^- are the sets of the positive and negative examples, respectively. Then, the following relation is true: $r \geq k(G) \geq \omega(\hat{G})$.

At this point it should be stated that according to this theorem the gap between r and $k(G)$ can be positive. The same is also true with the gap between $k(G)$ and $\omega(\hat{G})$. Therefore, there is a potential for the gap between r and $\omega(\hat{G})$ to be large (since the value of $\omega(\hat{G})$ can be arbitrarily large, see for instance [Golumbic, 1980] and [Bollobás, 1979]). Results from some related computational experiments in [Triantaphyllou and Soyster, 1996] seem to indicate that when the value of $\omega(\hat{G})$ is large, then the bound is rather tight.

Although finding $k(G)$ is NP-complete, the determination of $\omega(\hat{G})$ is also NP-complete, but there are more efficient enumerative algorithms. In Carraghan and Pardalos [1990] a survey of algorithms which can find the maximum clique in any graph is presented. They also present a very efficient algorithm which uses a partial enumeration approach which 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/3090-600S computer). Some other related developments regarding the maximum clique of a graph can be found in [Pardalos and Xue, 1994], [Babel and Tinhofer, 1990], [Babel, 1995], [Balas and Xue, 1993], and [Balas and Niehaus, 1994].

7. PROBLEM DECOMPOSITION

The rejectability graph provides an interesting framework for decomposing the determination of a lower bound for the number of clauses into a set of smaller problems. The decomposition is obtained through a partitioning of the rejectability graph. We consider two processes:

- *Decomposition via Connected Components, and*
- *Decomposition via the Construction of a Clique Cover.*

7.1 Connected Components

In this case, one inspects the rejectability graph for a *natural* decomposition. A *connected component* of a graph is a maximal subgraph in which there is a path of edges between any pair of vertices. Hence, the vertices of the connected components are mutually exclusive and their union is exhaustive. The following corollary is derived from Theorem 9 and illustrates the relation of the connected components of G and the clauses which can be inferred from two collections of positive and negative examples.

Corollary 1:
Suppose that E^+ and E^- are the sets of the positive and negative examples, respectively. Then, any subset of negative examples in E^- which is rejected by a single CNF 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.*

Pardalos and Rentala in [1990] present an excellent survey of algorithms which determine the connected components of a graph. Furthermore, they also propose a *parallel algorithm* which runs on an IBM ES/3090-400E computer (with four processors). That algorithm determines the connected components in super linear time.

The importance of Corollary 1 emerges when the sets of positive and negative examples are very large. First, one constructs the rejectability graph G. Next, one determines all the connected components of the rejectability graph by applying an algorithm (such as the one described in Pardalos and Rentala [1990]) for finding the connected components. Then, one solves 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 individual and distinct connected components in *G*.

In other words, if a graph has two or more connected components, then one can decompose the original problem into separate problems and *the aggregation of the optimal solutions (minimum number of CNF clauses) of the separate problems is an optimal solution to the original problem*. Observe that each such sub-problem (in the CNF case) is comprised of the negative examples for that component and *all* the positive examples, i.e. the positive examples are identical for each sub-problem.

7.2 Clique Cover

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 (i.e., they are covered).

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 yield 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 $\omega(\hat{G})$, 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 tests described in Section 8, provide some insight into the effectiveness of such decomposition approach.

The two problem decomposition approaches described in this section can be combined into one approach as follows. 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. This approach is further illustrated in the demonstrative example presented in the following section.

8. AN EXAMPLE OF USING THE REJECTABILITY GRAPH

Next we consider the following sets of positive and negative examples:

$$E^+ = \begin{bmatrix} 0&1&0&0&0&1&0&1&1&1\\ 0&1&1&1&1&1&0&0&0&0\\ 0&0&1&0&1&1&1&0&1&0\\ 0&1&0&0&1&1&0&1&1&0\\ 1&0&1&0&0&0&1&0&1&1\\ 1&1&1&0&0&0&0&0&1&1\\ 1&1&0&0&0&0&1&1&1&1\\ 1&0&0&1&0&0&1&1&0&1\\ 0&1&1&0&1&1&0&0&1&0\\ 0&0&1&1&0&1&1&0&0&1\\ 1&1&1&1&0&0&0&0&0&1\\ 1&0&1&0&1&0&1&0&1&0\\ 1&1&1&0&1&0&0&0&1&0 \end{bmatrix}, \text{ and}$$

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

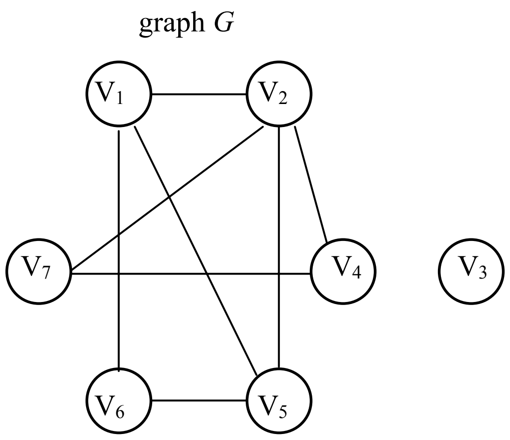

Figure 6. The Rejectability Graph for the New Sets E^+ and E^-.

One may use **any method** for inferring clauses from two disjoint classes of examples. An application of the OCAT approach in this illustrative example yields the following CNF system of four clauses:

$$(A_2 \vee A_3 \vee \bar{A}_5) \wedge (\bar{A}_1 \vee A_3 \vee \bar{A}_5) \wedge (A_1 \vee A_2 \vee A_4 \vee A_5) \wedge$$
$$\wedge\ (\bar{A}_1 \vee A_2 \vee \bar{A}_3 \vee \bar{A}_4).$$

Of course the question addressed in this section is whether it is possible to derive another system with *fewer* clauses.

To help answer the previous question, we apply Theorem 8 to this illustrative example. Since there are 13 positive and 7 negative examples, the construction of the rejectability graph requires 21 simple rejectability examinations. When Theorem 8 is applied to these data, the rejectability graph shown in Figure 6 is derived. For instance, there is an edge between vertices V_1 and V_6 because the first and sixth negative examples can be rejected by a single disjunction without violating the constraints imposed by the positive examples in E^+. A similar interpretation holds for the remaining edges in graph *G*.

The rejectability graph in the current illustrative example has *two connected components* (see also Figure 6). One component is comprised by the vertices V_1, V_2, V_4, V_5, V_6, V_7 and the second component has only the vertex V_3. Therefore, the original problem can be partitioned into *two independent* clause inference sub-problems.

Both sub-problems have the same positive examples. The first sub-problem has the same negative examples as in E^- *except* for the *third* negative example. The second problem has *only* the third negative example. The lower bound for the minimum number of CNF clauses required to appropriately classify the 20 examples is derived from the sum of the lower bounds for the two separate components. Since the rejectability graph of the second sub-problem contains only a single vertex, the size of the minimum clique cover is one. A minimum clique cover is also obvious for the first sub-problem, namely, the two sets $\{V_1, V_5, V_6\}$ and $\{V_2, V_4, V_7\}$. Hence, a minimum clique cover is two for the second sub-problem. Thus, an overall lower bound for the minimum number of CNF clauses required is three. Hence, it may well be possible that only three clauses are needed to appropriately classify all 20 examples.

As it was also mentioned in Section 2 of this chapter there is another clause inference approach which can be used to determine a minimum size set of clauses. This method, denoted as SAT (for satisfiability), has been proposed in Kamath, *et al.* [1992]. In that approach one first specifies an upper limit on the number of clauses to be considered, say k. That is, the value of k must be *pre-assumed*. Next a clause satisfiability (SAT) model is formed and solved using an interior point method developed by Karmakar and his associates [1992]. If the clause satisfiability problem is satisfied, 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 yields 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. In this illustrative example, the SAT approach with $k = 3$, is feasible and returns the Boolean function with the following 3 clauses:

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

However, when the value $k = 2$ is used, then the corresponding SAT formulation is infeasible. Therefore, this set of clauses is optimal in the sense of this chapter. The last statement also follows from Theorem 10 since there exists a clique cover of 3 and a set of clauses has been derived with exactly this number of members.

9. CONCLUSIONS

This chapter presented an approach for inferring a Boolean function from two classes of disjoint observations. The observations can be defined on multi-valued or binary valued attributes. A straightforward binarization approach is described as well. A minimization algorithm based on a branch-and-bound approach and a fast heuristic are also described.

A graph based approach for decomposing a large data mining problem into a series of smaller problems is described too. This graph based approach can also provide some bounds on the size of the inferred Boolean functions (when they are expressed in CNF or DNF format). A method for guided learning is also discussed.

Some of the results are specific to the proposed approach, termed OCAT (for One Clause At a Time) and other results can be combined with *any* data mining and knowledge discovery method. The presented methods have been tested on simulated and actual data as described in the cited papers with highly promising results.

Of particular interest are some extensions into *text mining* as described in [Nieto Sanchez, Triantaphyllou and Kraft, 2002]. Another interesting extension is the application of the OCAT approach to the mining of association rules [Yilmaz, Triantaphyllou, *et al.*, 2003]. In the later paper the application of a modified version of the OCAT approach significantly alleviates some computational problems that are caused by the huge number of the association rules that are usually returned by traditional methods. Finally it should be stated that some extensions into cases of having data with noise (stochastic data) seem to be possible with the use of monotone Boolean functions as discussed in Chapter 4 of this book authored by Torvik and Triantaphyllou [2005]. A recent book by the author [Triantaphyllou, 2005] describes in great detail all the previous issues, and much more, on data mining and knowledge discovery by means of a logic-based approach.

Future research in this area may be related to new ways for decomposing large size problems and also on the development of specialized methods for particular applications of data mining and knowledge discovery from databases. Another possible research direction might be the use of fuzzy logic and also on how to do all the above with multi-valued data directly without having to go through the binarization process first. Inferring Boolean functions from examples is a prominent area in data mining and knowledge discovery methods and more research in the future is almost guaranteed to be a hot area.

REFERENCES

Angluin, D., (1987), "Learning Propositional Horn Sentences With Hints," *Technical Report*, YALE/DCS/RR-590, Department of Computer Science, Yale University, Connecticut, U.S.A.

Babel, L. and G. Tinhofer, (1990), "A Branch and Bound Algorithm for the Maximum Clique Problem," *Methods and Models of Operations Research*, Vol. 34, pp. 207-217.

Babel, L., (1991), "Finding Maximum Cliques in Arbitrary and in Special Graphs," *Computing*, Vol. 46, pp. 321-341.

Babel, (1995), "A Fast Algorithm for the Maximum Weight Clique Problem," *Computing*, Vol. 10, pp. 12-23.

Balas, E. and J. Xue, (1993), "Weighted and Unweighted Maximum Clique Algorithms with Upper Bounds From Fractional Coloring," *Management Science Research Report #MSRR-590*, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A., 19 pages.

Balas, E. and W. Niehaus, (1994), "Finding Large Cliques by Bipartite Matching," *Management Science Research Report #MSRR-597*, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A., 11 pages.

Bartnikowski, S., M. Granberry, J. Mugan, and K. Truemper, (2005), "Transformation of Rational Data and Set Data to Logic Data," Chapter 7 in: "Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques," Triantaphyllou, E. and G. Felici (Eds.), Massive Computing Series, *Springer*, Heidelberg, Germany, pp. 253-278.

Bollobás, B., (1979), "Graph Theory, An Introductory Course, " *Springer*, Berlin, Germany.

Bongard, M., (1970), "Pattern Recognition," *Spartan Books*, New York, NY, U.S.A.

Boros, E., P.L. Hammer, and J.N. Hooker, (1994), "Predicting Cause-Effect Relationships from Incomplete Discrete Observations," *SIAM Journal on Discrete Mathematics*, Vol. 7, No. 4, pp. 531-543.

Bradshaw, G., R. Fozzard, and L. Cece, (1989), "A Connectionist Expert System that Really Works," *Advances in Neural Information Processing*, Morgan Kaufman, Palo Alto, CA, U.S.A.

Brayton, R., G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, (1985), "Logic Minimization Algorithms for VLSI Minimization," *Kluwer Academic Publishers*, Norwell, MA, U.S.A.

Brown, D., (1981), "A State-Machine Synthesizer-SMS," *Proceedings of the 18-th Design Automation Conference*, pp. 443-458.

Carraghan, R. and P.M. Pardalos, (1990), "An Exact Algorithm for the Maximum Clique Problem," *Operations Research Letters*, Vol. 9, No. 11, pp. 375-382 (1990).

Chang, I., R. Engel, D. Kandlur, D. Pendarakis, D. Saha, (1999), "Key Management for Secure Internet Multicast using Boolean Function Minimization Techniques," *Proceedings of IEEE Infocomm, 1999.* Also available as a PDF file from the Citeseer website.

Cohn, D., L. Atlas and R. Ladner, (1994), "Improving Generalizing with Active Learning," *Machine Learning*, Vol. 15, pp. 201-221.

Deshpande, A.S., and E. Triantaphyllou, (1998), "A Greedy Randomized Adaptive Search Procedure (GRASP) for Inferring Logical Clauses from Examples in Polynomial Time and some Extensions," *Mathematical and Computer Modelling,* Vol. 27, No. 1, pp. 75-99.

Dietterich, T.C., and R.S. Michalski, (1983), "A Comparative Review of Selected Methods for Learning from Examples," R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.). *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, U.S.A., pp. 41-81.

Fayyad, U.M., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, 996, Advances in Knowledge Discovery and Data Mining, *MIT Press,* Cambridge, MA, U.S.A.

Felici, G., and K. Truemper, (2002), "A Minsat Approach for Learning in Logic Domains," *INFORMS Journal on Computing,* Vol. 14, No. 1, Winter 2002, pp. 20-36 .

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.

Galant, S., (1988), "Connectionist Expert Systems," *Commun. of the ACM*, Vol. 31, No. 2, pp. 152-169.

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

Goldman, S., and R.H. Sloan, (1994), "The Power of Self-Directed Learning," *Machine Learning*, Vol. 14, pp. 271-294.

Golumbic, M.C., (1980), Algorithmic Graph Theory and Perfect Graphs, *Academic Press*, New York, NY, U.S.A.

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.

Hall, L., and A. Romaniuk, (1990), "A Hybrid Connectionist, Symbolic Learning System," *Proceedings of the AAAI '90*, Boston, MA, U.S.A., pp. 783-788.

Hattori, K. and Y. Torri, (1993), "Effective Algorithms for the Nearest Neighbor Method in the Clustering Problem," *Pattern Recognition*, Vol. 26, No. 5, pp. 741-746.

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

Haussler, D., (1988), "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework," *Artificial Intelligence*, Vol. 36, pp. 177-221.

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, U.S.A., pp. 291-312.

Hong, S., R. Cain, and D. Ostapko, (1974), "MINI: A Heuristic Approach for Logic Minimization," *IBM J. Res. Develop.*, pp. 443- 458.

Johnson, N., (1991), "Everyday Diagnostics: A Critique of the Bayesian Model," *Med. Hypotheses*, Vol. 34, No. 4, pp. 289-96.

Quine, W., (1952), "The Problem of Simplifying Truth Functions," *Am. Math. Monthly*, Vol. 59, pp. 102-111.

Quine, W., (1955), "A Way to Simplify Truth Functions," *Am. Math. Monthly*, Vol. 62.

Quinlan, J.R., (1986), "Induction of Decision Trees," *Machine Learning,* Vol. 1, No. 1, pp. 81-106.

Quinlan, J.R., (1979), "Discovering Rules by Induction from Large Numbers of Examples: A Case Study," D. Michie (ed.), *Expert Systems in the Micro-Electronic Age*. Edinburgh University Press, Scotland, UK.

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 36-th MSCAS*, pp. 1-5.

Kamgar-Parsi, B. and L.N. Kanal, (1985), "An Improved Branch-And-Bound Algorithm for Computing k-Nearest Neighbors," *Pattern Recognition Letters*, Vol. 3 pp. 7-12.

Karmakar, N.K., M.G.C. Resende, and K.G. Ramakrishnan, (1992), "An Interior Point Algorithm to Solve Computationally Difficult Set Covering Problems," *Math. Progr.,* Vol. 52, pp. 597-618.

Karnaugh, M., (1953), "The Map Method for Synthesis of Combinatorial Logic Circuits," *Transactions of the AIEE, Communications and Electronics*, Vol. 72, pp. 593-599.

Kearns, M., M. 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, J.F. Ruiz, V.I. Torvik, and E. Vityaev, (2000), "The Reliability Issue of Computer-Aided Breast Cancer Diagnosis," *Computers and Biomedical Research,* Vol. 33, No. 4, August, pp. 296-313.

Kurita, T., (1991), "An Efficient Agglomerative Clustering Algorithm Using a Heap," Pattern *Recognition,* Vol. 24, No. 3, pp. 205-209.

Mangasarian, O.L., W.N. Street, and W.H. Woldberg, (1995), "Breast Cancer Diagnosis and Prognosis Via Linear Programming," Operations *Research*, Vol. 43, No. 4, pp. 570-577.

Mangasarian, O.L., R. Setiono, and W.H. Woldberg, (1991), "Pattern Recognition Via Linear Programming: Theory and Application to Medical Diagnosis," Large-*Scale Numerical Optimization,* T.F. Coleman, and Y. Li, (Eds.), SIAM, pp. 22-30.

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

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

Motwani, R, and P. Raghavan, (1995), Randomized Algorithms, *Cambridge University Press,* 1995.

Nieto Sanchez, S., E. Triantaphyllou, J. Chen, and T.W. Liao, (2002), "An Incremental Learning Algorithm for Constructing Boolean Functions From Positive and Negative Examples," *Computers and Operations Research*, Vol. 29, No. 12, pp. 177-1700.

Nieto Sanchez, S., E. Triantaphyllou, and D. Kraft, (2002), "A Feature Mining Approach for the Classification of Text Documents Into Disjoint Classes," *Information Processing and Management*, Vol. 38, No. 4, pp. 583-604.

Pappas, N.L, (1994), Digital Design, *West Publishing Co.*, Minneapolis/St. Paul, MN, U.S.A.

Pardalos, P.M. and J. Xue, (1994), "The Maximum Clique Problem," *Journal of Global Optimization*, Vol. 4, pp. 301-328.

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, PA, U.S.A.

Peysakh, J., (1987), "A Fast Algorithm to Convert Boolean Expressions into CNF," *IBM Comp. Sci. 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.

Rivest, R.L., (1987), "Learning Decision Trees," *Machine Learning*, Vol. 2, No. 3, pp. 229-246.

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

Sun, R. and F. Alexandre (Eds.), (1997), "Connectionist-Symbolic Integration: From Unified to Hybrid Approaches," *Lawrence Erilbaum Associates, Publishers,* Mahwah, NJ, U.S.A.

Torvik, V.I., and E. Triantaphyllou, (2005), "Discovering Rules that Govern Monotone Phenomena," Chapter 4 in: "Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques," Triantaphyllou, E. and G. Felici (Eds.), Massive Computing Series, *Springer*, Heidelberg, Germany, pp. 149-192.

Towell, G., J. Havlic, and M. Noordewier, (1990), "Refinement Approximate Domain Theories by Knowledge-Based Neural Networks," *Proceedings of the AAAI '90 Conference*, Boston, MA, U.S.A., pp. 861-866.

Triantaphyllou, E., (2005), "Data Mining and Knowledge Discovery Via a Logic-Based Approach," Massive Computing Series, *Springer,* Heidelberg, Germany.

Triantaphyllou, E., and A.L. Soyster, (1996), "On the Minimum Number of Logical Clauses Which Can be Inferred From Examples," *Computers and Operations Research,* Vol. 23, No. 8, pp. 783-799.

Triantaphyllou, E., and A.L. Soyster, (1995a), "A Relationship Between CNF and DNF Systems Derivable from Examples," *ORSA Journal on Computing*, Vol. 7, No. 3, pp. 283-285.

Triantaphyllou, E., and A.L. Soyster, (1995b), "An Approach to Guided Learning of Boolean Functions," *Mathematical and Computer Modeling,* Vol. 23, No. 3, pp. 69-86.

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., 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.

Truemper, K., (2004), Design of Logic-based Intelligent Systems, *John Wiley & Sons, Inc.,* New York, NY, U.S.A.

Truemper, K., (1998), Effective Logic Computation, *Wiley-Interscience,* New York, NY, U.S.A.

Woldberg, W.W., and O.L. Mangasarian, (1990), "A Multisurface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology," *Proceedings of the National Academy of Sciences of the USA,* Vol. 87, No. 23, pp. 9193-9196.

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

Valiant, L.G., (1985), "Learning Disjunctions of Conjunctives," *Proceedings of the 9th IJCAI,* pp. 560-566.

Yilmaz, E., E. Triantaphyllou, J. Chen, and T.W. Liao, (2003), "A Heuristic for Mining Association Rules In Polynomial Time," *Mathematical and Computer Modelling,* No. 37, pp. 219-233.

AUTHOR'S BIOGRAPHICAL STATEMENT

Dr. Triantaphyllou did his graduate studies at Penn State University from 1984 to 1990. While at Penn State, he earned a Dual M.S. degree in Environment and Operations Research (OR), an M.S. degree in Computer Science and a Dual Ph.D. degree in Industrial Engineering and Operations Research. Since the spring of 2005 he is a Professor in the Computer Science Department at the Louisiana State University (LSU) in Baton Rouge, LA, U.S.A., after he has served for 11 years as an Assistant, Associate, and Full Professor in the Industrial Engineering Department at the same university. He has also served for one year as an Interim Associate Dean for the College of Engineering at LSU.

His research is focused on decision-making theory and applications, data mining and knowledge discovery, and the interface of operations research and computer science. Since the years he was a graduate student, he has developed new methods for data mining and knowledge discovery and also has explored some of the most fundamental and intriguing subjects in decision making. In 1999 he has received the prestigious IIE (Institute of Industrial Engineers), OR Division, Research Award for his research contributions in the above fields. Some of his graduate students have also received awards and distinctions including the Best Dissertation Award at LSU for Science, Engineering and Technology for the year 2003. In 2000 Dr. Triantaphyllou published a bestseller book on multi-criteria decision-making. Also, in 2005 he published a monograph on data mining and knowledge discovery, besides co-editing a book on the same subject.

He always enjoys sharing the results of his research with his students and is also getting them actively involved in his research activities. He has received teaching awards and distinctions. His research has been funded by federal and state agencies, and the private sector. He has extensively published in some of the top refereed journals and made numerous presentations in national and international conferences.

Dr. Triantaphyllou has a strong inter-disciplinary background. He has always enjoyed organizing multi-disciplinary teams of researchers and practitioners with complementary expertise. These groups try to comprehensively attack some of the most urgent problems in the sciences and engineering. He is a strong believer of the premise that the next round of major scientific and engineering discoveries will come from the work of such inter-disciplinary groups. More details of his work can be found in his web site (http://www.csc.lsu.edu/trianta/).