AN EXTENSION MATRIX APPROACH TO THE GENERAL COVERING PROBLEM

by

J. Hong
R. S. Michalski
C. Uhrik

In SPIE Applications of Artificial Intelligence III, Vol. 635, 1986.

An Extension Matrix Approach to the General Covering Problem*

Jiarong Hong**, Ryszard Michalski, and Carl Uhrik
Department of Computer Science, University of Illinois
1304 W. Springfield, Urbana, IL 61801

Abstract

A new approach, called the *extension matrix* (EM) approach, for describing and solving the *general covering problem* (GCP) is proposed. The paper emphasizes that the GCP is NP-hard and describes an approximately optimal covering algorithm, AE1. AE1 incorporates the EM approach with a variety of heuristic search strategies. Results show the new algorithm to be efficient and useful for large scale problems.

Introduction

An overview

The *general covering problem* (GCP) occurs often in various aspects of artificial intelligence, pattern recognition, switching theory, VLSI design, and other fields. This problem is of particular importance to machine learning and inductive inference[1].

The *general covering problem* (GCP) is an extension of the standard covering problem and is defined as follows. Let E be an *event space* spanning a finite set of *discrete-valued variables*. Certain subsets of the event space E are distinguished and called *complexes* (described in detail later). Given a partition of the event space into PE and NE, called *positive* and *negative event sets*, respectively, GCP is the problem of finding decision rules to classify groups of objects in a way that minimizes cost (e.g., rule length), where the *decision rules* are complexes whose set-theoretic union includes all PE and none of NE.

It is known that the general covering problem is NP-hard[2,3], and it is shown herein that many specializations of GCP are NP-hard[2]. Consequently, for solving complex problems, efficient approximate algorithms are of most interest. Michalski's A^q is such an efficient, quasi-optimal covering algorithm[3, 4, 5].

This paper introduces a structure, the *extension matrix*, for describing the general covering problem simply and constructing a new covering algorithm . Implementation shows that the new algorithm, AE1 (Extension Matrix Algorithm), is efficient and gives optimal or nearly optimal results. Moreover, AE1 handles large data sets. Experiments in the areas of bio-medical computing and used car dealerships demonstrate performance comparisons against various A^q algorithms and Rendell's probabilistic PLS1 [6] (which produces something akin to covers). This paper discusses in detail a variety of heuristic search strategies incorporated into AE1 to make it approximately optimal.

A simple example for GCP

Suppose there are two series of microprocessor systems: Mic_Non8080 and Mic_8080. A store owner wants simple rules for deciding the series of a system on the basis of such characteristics as RAM memory size, ROM memory size, display type, and number of keys on keyboard. Suppose the owner collected facts about the systems in stock and arranged the facts into Table 1. From this data, one can generate, for example, the following decision rules.

Mic_Non8080 ← [RAM=32K] v [ROM=10K v 80K]
Mic_8080 ← [RAM=48K v 64K][ROM=1K v 4K v 8K v 11_16K]

* This research was supported in part by the National Science Foundation under grant DCR 84-06801, Office of Naval Research under grant N00014-82-K-0186, Defense Advanced Research Project Agency under grant N00014-K-85-0878 and by a grant from the Education Ministry of the People's Republic of China.

**The author is a Visiting Scholar on leave from Harbin Institute of Technology, Harbin, People's Republic of China.

Mic_non8080			
RAM	ROM	Display	Keys
48K	10K	Color_TV	52
48K	10K	Color_TV	57_63
32K	11_16K	Color_TV	64_73
32K	80K	Built_in	92
32K	10K	B/W_TV	53_56
48K	10K	B/W_TV	53_56

Mic_8080			
RAM	ROM	Display	Keys
48K	4K	B/W_TV	57_63
64K	1K	Built_in	64_73
64K	8K	Terminal	57_63
48K	11_16K	B/W_TV	53_56
64K	8K	Built_in	64_73
48K	11_16K	Built_in	64_73

Table 1. Microprocessor Systems Data

Paraphrasing, the rules say if RAM size is 32K, or ROM size is 10K or 80K, then it is a non-8080 series system, but if RAM size is 48K or 64K, and ROM size is 1K, 4K, 8K, or within 11K to 16K, then it is a 8080 series. Expressing the example terms of Table 1, a characteristic (such as RAM or Keys) is a *variable*, an individual system is an *event*, and the given series is a *class*. If we were looking for Mic_Non8080s, a Mic_Non8080 system would be a *positive event*, while a Mic_8080 system would be a *negative event*. A statement such as [RAM = 48K v 64K] is a *selector*, and a rule such as [RAM = 48K v 64K][ROM = 1K v 4K v 8K v 11_16K] is a *complex*.

Terminology

This paper adopts the basic concepts of the *Variable-Valued system* VL_1 [5]. Table 2 gives a summary of the relevant notation used herein.

x_j	a *variable*, attribute or characteristic with values from set $\mathbf{D}_j$
$\mathbf{D}_j$	*domain of variable* x_j, a finite set of integers
$e=<v_1,\ldots,v_n>$	an *event* e, consisting of $v_j \in \mathbf{D}_j$, $j \in [1,n]$
v_j	the j^{th} *element* of event e.
PE	the *set of positive events*, the set to be covered
NE	the *set of negative events*, not to be covered
$\#(S)$	# is a function returning the cardinality of a set S
$\mathbf{A}$	a *reference set*, a subset of $\mathbf{D}_j$
$[x_j \neq \mathbf{A}]$	a *selector**, a statement that the value of x_j is *not* in $\mathbf{A}$
$[x_j = \mathbf{A}]$	an alternate *selector*, a statement that the value of x_j *is* in $\mathbf{A}$
$\&$	denotes conjunction, logical product
$L = \underset{j \in J}{\&} [x_j \neq A_j]$	A *complex* L, conjunction of n selectors

*Note: We use selectors mostly in form of $[x_j \neq A_j]$ rather than $[x_j = B_j]$, as the elements of a complex, where $B_j = \overline{A_j}$.

Table 2. Basic Symbols and Terms

Positive Events				
Event #	Variable #			
	1	2	3	4
1	0	0	0	0
2	0	0	2	0
3	0	2	0	1
4	0	1	1	0
5	0	2	2	1
6	0	2	1	0

(a) PE

Negative Events				
Event #	Variable #			
	1	2	3	4
1	0	2	1	1
2	0	0	3	0
3	1	2	0	0
4	1	1	1	0
5	1	0	2	1
6	1	2	3	0

(b) NE

Positive Event$_1$ EM				
Row #	e_1: 0 0 0 0			
	Elements			
1	*	2	1	1
2	*	*	3	*
3	1	2	*	*
4	1	1	1	*
5	1	*	2	1
6	1	2	3	*

(c) EM_1

Positive Event$_2$ EM				
Row #	e_2: 0 2 0 1			
	Elements			
1	*	*	1	*
2	*	0	3	0
3	1	*	*	0
4	1	1	1	0
5	1	0	2	*
6	1	*	3	0

(d) EM_2

Positive Event$_4$ EM				
Row #	e_4: 0 1 1 0			
	Elements			
1	*	2	*	1
2	*	0	3	*
3	1	2	0	*
4	1	*	*	*
5	1	0	2	1
6	1	2	3	*

(e) EM_4

Table 3 – Positive and Negative Events with Extension Matrices.

The following definitions give the condition under which a complex covers a positive event in PE and does not cover any negative event in NE.

Def 1.1: Given an event, $e = <v_1, \ldots, v_n>$, and a complex, $L = \underset{j \in J}{\&} [x_j \neq A_j]$, L is said to *cover* e if for each $j \in J$, $v_j \notin A_j$. L *does not cover* e if there exists at least one j such that $j \in J$ and $v_j \in A_j$.

Def 1.2: A complex L *covers e against NE* if L covers e and does not cover any event in NE.

Def 1.3: A *cover of class PE against class NE* is a set of those complexes that cover PE and do not cover any event in NE.

In the microprocessor example, complex [RAM ≠ 16K v 48K v 64K], equivalent to [RAM = 32K], covers three events (the third, fourth, and fifth) in Mic_Non8080 since the element 32K of the three events is absent in the complex. Also, the complex does not cover any event in Mic_8080 since, for example, the first event in Mic_8080 has an element 48K which appears in the values to be excluded from the complex. Similarly, [ROM = 10K v 80K] covers the remaining events in class Mic_Non8080, and complex [RAM = 48K v 64K][ROM = 1K v 4K v 8K v 11_16K] covers class Mic_8080 against Mic_Non8080.

The extension matrices

This section introduces a basic concept, the extension matrix, on which the covering algorithm AE1 is based. Suppose PE and NE are the sets of the positive and negative events, as shown in (a) and (b) of Table 3.

Def 2.1: Given a positive event $e_i = <v_1, \ldots, v_n>$ in PE, if one substitutes * (referred to as the *dead element*) for all appearances of v_j in the j^{th} column of NE, for $j \in [1,n]$, the resulting matrix is called the *extension matrix* of event e_i against NE, denoted by EM_i.

Def 2.2: A set of m non–dead elements that come from m different rows is called a *path*, where m = #(NE).

Def 2.3: Corresponding elements in different extension matrices that have the same value are said to be a *common element* of the matrices, and a path that consists only of common elements is said to be a *common path* of the extension matrices involved. Two extension matrices are *disjoint* if they share no common path.

For example, (c), (d), and (e) of Table 1 are the extension matrices of events e_1, e_3, and e_4 respectively. Letting r_{ij} denote an element r in the i^{th} row and the j^{th} column in EM_1, path 1_{13}-3_{23}-1_{31}-1_{41}-1_{51}-1_{61} in (c) stands for the complex L $= [x_1 \neq 1][x_3 \neq 1,3]$, or equivalently $[x_1=0][x_3=0,2]$. This path is also a common path of EM_1 and EM_3. Also, EM_3 and EM_4 are disjoint since in the first row they have no common element.

From definitions 1.1-2.3, one can prove the following theorem which shows a one-to-one mapping ϕ from the set of the paths in an extension matrix to the set of complexes that cover e against *NE*. Thus, a path and the corresponding complex may be considered as the same thing.

Theorem 1: Let EM_e be an extension matrix of e against *NE*, then each path represents a complex that covers e against *NE*, and each such a complex corresponds to a path.

Proof: Suppose $P = < r_{1j_1}, \ldots, r_{mj_m} >$ is a path in EM_e, then a complex, say L, that consists of r_{ij_i}, for i = 1 ... m and $j_i \in [1,n]$, covers e against *NE* by the following argument. Since no element of e is the same as r_{ij_i}, i = 1 ... m, so L covers e. Also, for each i, r_{ij_i} is an element of negative event $\bar{e}_i$, so L does not cover $\bar{e}_i$. Conversely, given a complex that covers e against *NE*, a path can be structured in a way that the path consists of such elements of the complex that are in EM_e. Also, for each row i in EM_e, there exists an element that is in the path, since otherwise the negative event $\bar{e}_i$ would be covered by the complex.

Theorem 1 shows that an EM_e contains the paths which correspond to all complexes that cover e against *NE*.

Def 9: The *star* G of *e* against *NE* is the set of all complexes that cover *e* against *NE*, denoted by G(e | *NE*). The event *e* is called the *seed of the star.*

Note that a star of seed e is isomorphic to the corresponding extension matrix EM_e.

NP-Hard problems in the general covering theory

The optimal set covering problem (SETCV) is as follows: Given a finite cover of a finite set, find the subcover which uses the fewest sets from the given cover. More precisely, suppose T is a set of m points and the given cover F is a finite family $\{s_1, s_2, \ldots, s_p\}$, such that $\cup s_i = T$ and $p \leq m$, find a solution $F' = \{ s_j \mid s_j \subseteq F \text{ and } \cup s_j = T \}$ so that #(F') is minimal.

The following are some optimization problems in the GCP which are shown by Hong and Michalski[2] to be NP-hard.

(I) MCV: Generating a cover that has the minimum number of complexes.

(II) MCOMP: Generating a complex which has the minimum number of selectors; such complexes are called *minimal complexes.*

(III) MSCV: Generating a cover which consists of only minimal complexes.

(IV) MCVS: Generating a cover that has the minimum number of complexes and consists of only minimal complexes.

(V) MINF: Generating the minimum complete family of extension matrices (i.e., generating the minimal set of EMs which contain complexes covering all *PE* and none of *NE*).

As an illustration of the type of reasoning required to justify such theorems, we give an example from (I). Let T be a set of 9 points and let $F = \{ S_j \}$ be the given cover of the SETCV problem depicted in Table 4a. It is known that SETCV is NP-hard[7], so showing SETCV reduces to MVC in polynomial time suffices to verify that MCV is NP-hard. The *characteristic matrix* F, shown in Table 2b, may be analogously considered to be a set of positive events for a GCP whose sole negative event is that of Table 4c. From the positive and negative events, the extension matrices of Table 4d are derived. Since there is only one negative event and its elements all are 0, each EM consists of only one row and each of its non-dead elements (the 0 elements) are complexes. Corresponding elements in two or more EM are equal to 0 and are the common complexes of the EM. There is an exact correspondence between the complexes of the MCV problem and the sets of the SETCV problem, such that finding a minimum complex cover does indeed generate the smallest set subcover. Similarly, any SETCV problem can be reduced to an MCV problem, and thus MCV is NP-hard.

The extension matrices algorithm AE1

According to Theorem 1, one can directly construct a covering algorithm to generate a cover of *PE* against *NE* by appropriate choice of paths in corresponding extension matrices. Note that a complex generated this way may cover several events. For instance, the complex L in Table 3 covers four positive events, e_1, e_2, e_3, and e_5, so the actual number of complexes generated is smaller than the number of the positive events. Furthermore, the appropriate search strategies can

$F = \{S_1, S_2, \ldots, S_7\}$, where $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$S_1 = \{1, 3, 5\}$, $S_2 u = \{1, 4, 5\}$, $S_3 = \{2, 3, 7\}$, $S_4 = \{1, 2, 4, 6\}$, $S_5 = \{3, 4, 5\}$, $S_6 = \{3, 5, 7, 8\}$, $S_7 = \{2, 6, 7, 8, 9\}$.

Sets							
Point #	S_1	S_2	S_3	S_4	S_5	S_6	S_7
1	(1)	1		(1)			
2			2	(2)			(2)
3	(3)		3		3	3	
4		4		(4)	4		
5	(5)	5			5	5	
6				(6)			(6)
7			7			7	(7)
8						8	(8)
9					9		(9)

(a) Sets for SETCV

Negative Events (NE)							
Event #	0	0	0	0	0	0	0

(c) Negative Event for MCV

Characteristic Matrix (F)							
Point #	$F_{ij} = 1$ if $P_i \in S_j$; $F_{ij} = 0$ if $P_i \notin S_j$						
	F_{i1}	F_{i2}	F_{i3}	F_{i4}	F_{i5}	F_{i6}	F_{i7}
1	1	1	0	1	0	0	0
2	0	0	1	1	0	0	1
3	1	0	1	0	1	1	0
4	0	1	0	1	1	0	0
5	1	1	0	0	1	1	0
6	0	0	0	1	0	0	1
7	0	0	1	0	0	1	1
8	0	0	0	0	0	1	1
9	0	0	0	0	1	0	1
Event #	Positive Events (PE)						

(b) Characteristic Matrix for SETCV and Positive Events for MCV

Extension Matrices							
EM_k	x_1	x_2	x_3	x_4	x_5	x_6	x_7
1	(0)	0	*	(0)	*	*	*
2	*	*	0	(0)	*	*	(0)
3	(0)	*	0	*	0	0	*
4	*	0	*	(0)	0	*	*
5	(0)	0	*	*	0	0	*
6	*	*	*	(0)	*	*	(0)
7	*	*	0	*	*	0	(0)
8	*	*	*	*	*	0	(0)
9	*	*	*	*	0	*	(0)

(d) Extension Matrices and Complexes $[x_i \neq 0]$ for MCV

(a) sets ======> (d) EM

SETCV = $\{s_1, s_4, s_7\}$: <======> MCV = $\{ [x_1 \neq 0], [x_4 \neq 0], [x_7 \neq 0] \}$.

Table 4. SETCV is reduced to MCV .

substantially optimize the cover to be generated. Now, comes an outline of algorithm AE1, including some strategies it incorporates.

Generating complexes

When implementing AE1, it is unnecessary to build the actual extension matrices. Suppose $e = <e_j>$, $j = 1 \ldots n$, is a positive event to be covered, and $NE = <\bar{e}_{ij}>$, $i = 1 \ldots m$, is the matrix of negative events. In order to generate a complex L that covers e against NE, AE1 selects an element r_i from each row of NE such that $r_i \neq e_j$, where the inequality guarantees element r_i is not a dead element. Also, the complexity of generating a complex is at most $n*m$.

Paired associate searching

In order to limit the choice of elements in generating complexes, AE1 does *paired associate searching,* finding a common path from two extension matrices. The following theorem gives the condition under which two extension matrices are disjoint, that is, have no common path.

Theorem 2: Given the extension matrices, EM_k and EM_l, of two events $e_k = <e_{kj}>$ and $e_l = <e_{lj}>$, then
- an element r_{ij} in EM_k is a common element if and only if $r_{ij} \neq e_{kj}$ and $r_{ij} \neq e_{lj}$.
- EM_k and EM_l are disjoint if and only if there is at least one row, say the i^{th}, such that, for all j, $r_{ij} = e_{kj}$ or $r_{ij} = e_{lj}$.

Proof: It suffices to note that the elements in EM_k and EM_l are the same as those in NE except for the dead elements.

The optimization criteria and evaluation matrices

AE1 optimizes a cover according to the following three criteria: 1) *minimizing the number of complexes,* 2) *minimizing the number of selectors,* and 3) *minimizing the number of complexes and selectors.* In order to achieve criterion 1, the complexes generated need to cover as many events as possible, that is, the corresponding paths are the common paths of as many extension matrices as possible. For criterion 2, the number of columns involved for each complex needs to be as small as possible, that is, the columns that have the fewest dead elements are the candidates for searching. Criterion 3 is a combination of criterion 1 and 2. In accordance with these three criteria, we build three *evaluation matrices.* The first consists of those elements which are the number of appearances of each element of positive events in PE, as shown in (a) of Table 5. The second consists of the appearances of each element of negative events in NE, as shown in (b). The third consists of the quotients of the elements of matrix (a) divided by the corresponding elements of matrix (b), or by 0.1 if the corresponding elements are equal to 0, as shown in (c) (for class 1 (PE)) or (d) (for class 2 (NE)).

The costs of variables

We can see that in the evaluation matrix (a), the larger the element is, the more positive events there may be which share the common element. Thus, AE1 first sorts all the elements in the matrix in descending order, then assigns a cost to each positive event, according to the sequence of appearance of the elements in the sorted matrix. AE1 searches a path according to the cost of elements in the positive event from largest to smallest. For example, in Table 1 (a), e_4, $<0,1,1,0>$, corresponding to the numbers of appearances $<6,1,2,4>$ and has cost ordering (1,4,3,2), so AE1 will search the path in the order variable 1, variable 4, variable 3, then variable 2. Similarly, the processing applies to (b) with ascending order and to (c) or (d) with descending order.

Selecting seeds

Appropriate seed selection can improve results. In practice, according to the criteria, AE1 rearranges the set PE of positive events in such a way that: events which contain the appearances of the first element in the sorted evaluation matrix are put in the first place, those which contain the appearance of the second element in the second place, and so on. AE1 also refines the cover generated by repeating seed selection a number of times.

	Variables			
Values	1	2	3	4
0	6	2	2	4
1	0	1	2	2
2		3	2	
3			0	

(a) For class 1 (using criterion 1)

	Variables			
Values	1	2	3	4
0	2	2	1	4
1	4	1	2	2
2		3	1	
3			2	

(b) For class 1 (using criterion 2)

	Variables			
Values	1	2	3	4
0	3.0	1.0	2.0	1.0
1	0.0	1.0	1.0	1.0
2		1.0	2.0	
3			0.0	

(c) For class 1 (using criterion 3)

	Variables			
Values	1	2	3	4
0	0.33	1.0	0.5	1.0
1	40.0	1.0	1.0	1.0
2		1.0	0.5	
3			20.0	

(d) For class 2 (using criterion 3)

Table 5 – The Evaluation Matrices

Partitioning *PE* into subsets

AE1 will refine each cover of classes by partitioning the set of positive events into three disjoint subsets, selecting seeds independently, and generating a cover for each subset. In AE1, *PE* is partitioned using the experimental function bound(i) as the size of the partition, where i is the number of complexes generated for the current subset. The value b = bound(i) indicates that among the i complexes generated, only the first b complexes which cover the maximum number of events are chosen.

Learning

When applying a strategy, by repetition and comparison, AE1 chooses the best results. For example, if criterion = 1, then for each class, AE1 runs twice according to the combinations of criterion 1 and 2, and criterion 1 and 3, then by applying the set-covering algorithm, described in the next section, AE1 generates the third cover from the first two covers generated, finally, AE1 chooses the best one among the three covers generated. Thus, AE1 has learning ability throughout the searching.

The incorporation of the set-covering algorithm

We incorporate the approximate set-covering algorithm (c1) by Johnson[7] into AE1 to select a more desirable cover as previously mentioned (Partitioning *PE* into Subsets). The algorithm works as follows. For each pass among the complexes generated, AE1 chooses the one which covers the maximum number of events, and deletes the covered events. Then AE1 repeats the processing until set *PE* of positive events becomes empty.

The incorporation of deductive inference

Some deductive inference rules in mathematical logic were incorporated into AE1 for rewriting and simplifying the decision rules generated. The rules used in AE1 are the following.

Suppose a is a selector, say $a = [x = A]$, $\neg a = [x \neq A]$, and P, Q, R, and S are complexes. Let $\neg$, &, v, and ::> be negation, conjunction, disjunction, and implication linking a concept description with a concept name [8]. Then we have the following deductive inference rules.

(I) If a v a&P ::> K, then a ::> K. (II) If a v ¬a&P ::> K, then a v P ::> K.

(III) If a&P v ¬a&P ::> K, then P ::> K. (IV) If P&Q&S ::> K and P&Q ==> R then R&S ::> K.

Formulas (I) and (III) can be used to optimize the covers, and formula (II) and (IV) can be used to optimize the complexes.

Generating maximal complexes

Finally, AE1 always generates *maximal complexes*. Maximal complexes are those with all redundant selectors removed. For example, as shown in (e) of Table 3, complex $L = [x_1 \neq 1][x_2 \neq 2][x_3 \neq 3][x_4 \neq 1] = L'[x_4 \neq 1]$, which corresponds to path 2_{12}-3_{23}-1_{31}-1_{41}-1_{54}-3_{63}, contains a redundant selector $[x_4 \neq 1]$. If we delete the redundant selectors, then the *subexpressions* that cannot be simplified further are maximal complexes. Now, we give the formal definitions of maximal complex and a relevant concept.

Def 4.1: A complex L that covers e against *NE* is a *maximal complex* if it contains no proper subexpression that also covers e against *NE*.

Def 4.2: Two elements in an EM are *similar* if they are in the same column and have the same value. An element v is termed a *brother* of the other v′ if they are in the same row. In a path, an element v is called a *redundant element* of the path if it and all its similar elements have brothers such that the brothers or their similar elments are in the path.

For example, in (e) of Table 3, all elements 1 in the first column are similar to each other, 1_{51} is a brother of 2_{53}, and 1_{54} is a redundant element of the path drawn since its brother 1_{51} is similar to 1_{31} in the path, and its similar element 1_{14} has a brother 2_{12} in the path. Now we give the condition under which a complex is maximal.

Theorem 3: A complex L that covers e against *NE*, where $L = \&[x_j \neq A_j]$, is maximal if and only if the corresponding path has no redundant elements.

Proof: Suppose L is maximal, but there exists a redundant element, say v1, in the corresponding path P, that is, $L = L' \ \& \ [x_1 \neq v1]$. Since v1 and all its similar elements already have brothers in P, we can delete v1 and its similar elements from P. The resulting subexpression L′ still covers e against *NE*, which contradicts the fact that L is maximal. Conversely, suppose P has no redundant elements, but L is not maximal. There exists a subexpression L′ such that L′ covers e against *NE*. Then, $L = L' \ \& \ L''$, where L" is not the empty complex. We see that any element, say v_{ij}, in L" must have a brother in the path P′ corresponding to L′ since otherwise L′ would cover the negative event $\bar{e}_i$, which contradicts the fact that L′ covers e against *NE*. Thus, any element in L" is a redundant element of P, which is also a contradiction.

In Table 3, the complex in EM_1 is a maximal complex, but the complex in EM_4 is not maximal. Since generating maximal complexes can optimize a complex, AE1 has a procedure to generate maximal complexes.

Experiments and performance summary of AE1

AE1 run on a Pyramid machine and on a VAX 780 using some sample data sets gave optimal or near optimal results. For example, for a data set with 1236 events and 11 variables, available result was obtained, as shown in Table 4. Specifically, AE1 can handle a large data set containing 2^{15} events for simplifying switching circuits and gave more desirable results than those given by the VLSI design tool EQNTOTT[9]. Considering the complexity of generating a complex (previously mentioned), one concludes AE1 is such an efficient covering algorithm that its worst case time complexity is determined only by the product of the following items:

1. The number of given classes,
2. The number of variables,
3. The number of positive events on an average,
4. The number of negative events on an average.

A set of 4 experiments demonstrated the comparative performance of AE1 against 3 other programs producing covers. The additional algorithms compared are GEM (a successor of AQ11), AQ15 [10] (the latest AQ program), and PLS1. It is noted that PLS1 *regions* were taken to be complexes without any loss of generality, but not all complexes are representable as regions. It is also noted that PLS1, being a probabilistic algorithm, permits an error bound, and hence, the PLS1 covers contain errors whereas the other algorithms guarantee no errors.

Experiment 1 is an attempt to learn rules for distinguishing 6 human sleep stages[9]. There are 1236 events composed of 11 features. Each event is a vector representing a 30 second interval of real time, reduced from 5 channels of data (2 EEG, 1 Electromyograph, 2 Electro-oculograph). Experiment 2 is the same as 1, but the event set has been reduced. Experiment 3 derives rules for distinguishing 3 types of regular heart arrhythmias[10]. There are 263 examples of ECG recordings which are described in terms of 7 attributes. Experiment 4 is toy problem wherein a car dealer has three manufacturers of pickup trucks: Ford, Dodge, Chevy, and he wants simple rules for deciding the manufacturer of a truck on the basis of such characteristics as length in feet, exterior and interior color, and number of passengers. Hence, there are 4 attributes and 27 events. Table 6 gives a comparison among the known covering algorithms for these experiments.

Data Set	EXP 1	EXP 2	EXP 3	EXP 4
# Events	1236	621	263	27
# Variables	11	11	7	4
# Values (ave.)	9.1	9.1	5.9	12.3
# Class	6	5	3	3
Algorithm	#Cpx : Time	#Cpx : Time	#Cpx : Time	#Cpx : Time
AE1	85 : 47.4	36 : 20.4	6* : 0.8	5* : 0.3
GEM	95 : 305.6	41 : 43.9	6* : 1.5	5* : 0.3
AQ15	61 : 330.4	41 : 46.9	6* : 1.7	5* : 0.3
PLS1	75 : 187.4	38 : 68.1	13 : 8.6	9 : 2.3

*Note: The results marked by * are optimal.

Table 6 – Performance Summary (*Note: Time is in CPU seconds.)

Conclusions

We have introduced the extension matrix method to the general covering problem, and presented an efficient approximately optimal covering algorithm, AE1. Also, we have analyzed some heuristic strategies which were incorporated into AE1. We conjecture that incorporating more powerful strategies into AE1 will make it more efficient and more nearly optimal.

Acknowledgments

We would like to thank Robert Stepp, Igor Mozetic, Gail Thornburg, Stephen Borodkin, Tom Channic, Anthony Nowicki, Jeff Becker and Robert Reinke for their many helpful suggestions throughout the course of this research. We also thank Sylvian Ray for access to the sleep stage experiment data. The work presented here was done at the Artificial Intelligence Laboratory of the University Illinois, Department of Computer Science. The research was supported in part by the National Science Foundation (grant DCR 84-06801), the Office of Naval Research (grant N00014-82-K-0186), Defense Advanced Research Project Agency (grant N00014-K-85-0878), and the Education Ministry of the People's Republic of China.

References

1. Michalski, R. S., and Chilausky, R. L., "Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in context of developing an expert system for soybean disease diagnosis," *Policy Analysis and Information Systems,* Vol. 4, No. 2, pp. 125-160, June 1980, (Special issue on knowledge acquisition and induction).

2. Hong. J. R., and Michalski, R. S., "Some NP-Hard Problems and an Approximate Method in the General Covering Theory," to appear in Reports of the Intelligent Systems Group, Dept. of Computer Science, Univ. of Illinois, 1986.

3. Michalski, R. S., and McCormick, B. H, "Interval Generalization of Switching Theory," *Proceedings of the Third Annual Houston Conference on Computer and System Science*, Houston, Texas, April 26-27, 1971.

4. Michalski, R. S., "On the Quasi-Minimal Solution of the General Covering Problem," *Proceedings of the Fifth International Symposium on Information Processing* (FCIP 69), Vol. A3 (Switching Circuits), Bled, Yugoslavia, October 8-11, 1969.

5. Michalski, R. S., "A Variable-Valued Logic and its Application to Pattern Recognition and Machine Learning," *Multiple-Valued Logic and Computer Science*, D.(Ed.), Rine, North-Holland, pp. 506-534, 1975.

6. Rendell, L. A., "Genetic Plans and the Probabilistic Learning System: Synthesis And Results," Dept. of Computer Science, Report No. UIUCDCS-R-85-1217, Univ. of Illinois, 1985.

7. Johnson, D. S., "Approximation algorithm for combinatorial problems," *Journal of Computer and Systems Sciences* 9:3, April 30,- May 2, 1973, pp. 38-49.

8. Michalski, R. S., "A Theory and Methodology of Inductive Learning," *Machine Learning*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, (Ed.), Tioga Publishing Company, Palo Alto, CA, pp.83-135, 1983.

9. Mayo, R. N., Ousterhout, J. K, and Scott, W. S., "1983 VLSI Tools: Selected Works By The Original Artists," Computer Science Division (EECS), University of California, Berkeley, Report No. UCB/CSD 83/115, March 1983.

10. Hong, J., Mozetic, I., and Michalski, R., "AQ15: Learning Attribute-Based Descriptions from Examples, Algorithm and User's Guide," Dept. of Computer Science, Report No. UIUCDCS-F-85-949, Univ. of Illinois, 1986.

11. Ray, S. R., Lee, W. D., Morgan, C.D. and Airith-Kindree, W., "Computer Sleep Stage Scoring — An Expert System Approach," Dept. of Computer Science, Report No. UIUCDCS-F-85-943, Univ. of Illinois, 1985.

12. Mozetic, Igor, "Compression of the ECG Knowledge-base Using the AQ Inductive Learning Algorithm," Reports of the Intelligent Systems Group, File No. UIUCDCS-F-85-943, Dept. of Computer Science, Univ. of Illinois, 1985.