DECOMPOSITION OF RELATIONS: A NEW APPROACH TO CONSTRUCTIVE INDUCTION IN MACHINE LEARNING AND DATA MINING - AN OVERVIEW

Marek Perkowski, Dept. of Electrical and Computer Engineering,
Portland State University, Portland, Oregon 97207,
Tel: 503-725-5411, Fax: 503-725-4882, mperkows@ee.pdx.edu
Stanislaw Grygiel, Intel Corporation, Hillsboro, Oregon

Abstract

This is a review paper that presents work done at Portland State University and associated groups in years 1989 - 2001 in the area of functional decomposition of multi-valued functions and relations, as well as some applications of these methods.

I. INTRODUCTION

The data collected in many research, business, medical, manufacturing and robotics domains contain valuable information. Since the volume of the data is often large (the number of factors (features) it depends on may be on the order of 10^2 or even 10^3 in some medical applications and the number of records on the order of 10^9) the problem of automatic data analysis and information retrieval becomes more important than ever.

Knowledge Discovery in Databases (KDD) and **Data Mining** is the research area focused on problems of automating data analysis and information retrieval from large data sets. While KDD usually refers to the overall process of the data analysis and information retrieval, including selection, preprocessing and transformation of data, data mining refers to the very process of application of specific algorithms for extracting useful information from the data. The data mining algorithm is any algorithm that fits hypotheses to data or enumerates patterns from data [21].

Learning general concepts from limited number of instances of data is the domain of inductive inference. It consists of generating hypotheses, fitting them to the data and selecting the one(s) which best describe the data. To represent hypotheses, learning system can use a variety of models, such as linear discriminator, nearest neighbor classifier, decision tree, neural network, etc. Learning process consists on adjusting model parameters according to certain performance criteria. There is no universally best model for all possible problems. For instance, Bayes' classifiers seem to provide good results in medical domains [34], neural networks are likely to perform well in parallel domains, while decision diagrams are better when dealing with sequential domain problems [63]. The selection of the model depends on the type of the problem and constitutes the learning bias.

Machine learning is a process of examination of classified data cases in order to build a classifier that would correctly classify future data for which class assignments are not known. The quality of the classifier is usually scored based on it's predictive accuracy. In some domains however, predictive accuracy is not the only measure of interest for the users. What is often more important is interpretability of rules encoded in the classifier.

The interpretation of the results is especially important in medical applications where not only correct decision is desired but also explanation why that particular decision has to be made. That is why systems which provide human understandable results like decision diagrams are considered to be more acceptable than "black box" classifier models like neural networks. It is not because the former provide more accurate decisions than the later, it is because the decision process can be easily followed and understood in one case and is hidden to the user in the other.

The methods following these criteria can be used for any application requiring development of human understandable rules in technical, medical and other domains. Since 1989 our group has been working on a new method based on **decomposition** as a way of simplification of the description of data. It follows the general **Occam Razor principle**, restating in the present context, that simpler description of data has better generalization properties. The decomposition type is the well known (from the logic synthesis area) Ashenhurst-Curtis [14] simple serial decomposition and its modifications [27] and generalizations [69, 70, 47] but its application to **decomposition of relations** and **constructive induction** in Machine Learning and Data Mining is novel. Our decomposition programs implementing these ideas were the **first decomposer** being able to decompose large, incompletely specified, non-noisy and noisy functions and relations, common in machine learning applications (medical among others)[50, 52, 56, 23, 29, 27]. In the framework of the decomposition process, methods for effective **data reduction** (removing vacuous variables) have been developed to further simplify data description and speed up the decomposition process [29, 46].

Learning Bias

Learning process is always biased. Learning systems are biased too. We make assumptions and choices when building learning systems, implementing them and adjusting their parameters. These are all biases. Learning biases can be divided into two major categories: model selection bias and hypothesis selection bias [15].

Model Selection Bias

To build a classifier, a model for it has to be selected. The choice depends on the problem the classifier will be used for, and includes, but is not limited to, factors like simplicity of implementation, speed, accuracy, and personal preferences of the designer. The list of possible choices includes Linear Discriminators, Nearest Neighbor Classifiers, Decision Trees [61, 62, 63] and Diagrams, Neural Networks, Feature Vectors, First Order Logic and many others.

Hypothesis Selection Bias

Within a framework of a given classifier model many hypotheses or theories may be formulated. The choice of the best one is often a difficult task. To facilitate this task, selection criteria have to be determined which are simple and robust enough to make hypothesis selection an efficient process. The list of possible choices includes for instance Principle of Multiple Explanations, Occam's razor principle, and PAC learning [71, 76].

In the last few years incompletely specified, multiple-valued relations (functions in particular) are becoming increasingly important in image processing, machine learning, knowledge discovery, data-base optimization, artificial intelligence, image coding, automatic theorem proving and verification [51]. The appropriate representation of such relations is very important as it not only allows for storing larger functions, but also, to carry efficiently appropriate calculations. For instance, success of many **binary** decomposers depends on appropriate innovative representation of Boolean functions: cube calculus [73], spectral transforms [68], decision diagrams [35], [66], or rough partitions [42]. Good representations for large multiple-valued relations, both completely and incompletely specified, which would allow for representing uncertainties is essential for effective storing and processing of the type of data that are taken from practical technical and medical domains.

II. Generalized MV values and Relations

A multiple-valued (MV) directed relation (or function), can be in general defined by a set of feature vectors (Table 1). Each vector element is either a number or a set of numbers, or '–' symbol. The meaning of each is the following:

- A *number* represents MV variable value.
- A *set of numbers* represents generalized MV value. **Generalized MV value** refers to the situation where a variable can take any value from a subset of all possible values for that variable. For instance, 0,3 corresponds to the situation where a variable can take either 0 or 3 value (but not both).
- '–' symbol is a special case of generalized MV value where the set of values the variable can take is equal to the set of all possible values for that variable. The symbol comes from the binary logic where it represents don't care $\{0, 1\}$ value.

Generalized MV values are used in the Table 1 to compress the data. For instance vector #2 (1 1 0 0 0 1,2) is equivalent to two vectors (1 1 0 0 0 1) and (1 1 0 0 0 2). Some variables can be designated to be output variables and the relation becomes **a directed relation** expressing relationship between independent (input) and dependent (output) variables. If the output variable(s) can take only single values (generalized MV value cardinality is equal to 1) then the directed relation becomes a function (for instance a in Table 1 selected to be the output variable).

vector #	a	b	c	d	f	g
0	0	0	0	0	0,1	-
1	0	1	1	0,1	1,2	0,1
2	1	1	0	0	0	1,2
3	0	0,2	0	1	0,3	0,2
4	1	1	2	1	0,3	0
5	1	0	-	1	0,4	1

Table 1: Four-variable relation

In a common classification task a data sample needs to be assigned to one of the existing classes; data sample needs to be classified. Let us assume that there are three classes the input sample can be assigned to: 'a car', 'a tank', and 'an airplane', denoted as MV values $\{0\}$, $\{1\}$ and $\{2\}$, respectively. Let us also assume that expert decision is: 'I do not know, but not an airplane'. Now the question is: how to represent expert's uncertainty without loosing any information? The answer is generalized MV value, {a car or a tank}, denoted by $\{0, 1\}$. The concept of generalized MV values has various applications in Machine Learning and Knowledge Discovery from Databases. For instance, several ML benchmarks from Machine Learning benchmarks repository at U.C. Irvine, are relations: *hayes, flare1, flare2*. Relations have also been used in multiple-level logic design [75] and design of non-deterministic state machines. An example of non-deterministic state machine is a three-state binary counter realized with two flip-flops. There are four possible combinations of flip-flop outputs: {00,01,10,11}. If the counting order is $00 \rightarrow 01 \rightarrow 10 \rightarrow 11$ and we want to have a transition from state 11 to any state of the counter but not to 11, then the transition from state 11 is described by a generalized MV value $\{00, 01, 10\}$. The generalized MV values can also be used for discretization of continuous variables that occur in many ML applications. For instance, if we want to discretize a continuous value 3.5 we have a choice between selecting values 3 and 4. The best approach seems to be keeping both values ($\{3, 4\}$ generalized MV value) and postpone the selection of one of them until more information is available (principle of indifference).

III. Decomposition approach to Constructive Induction

A. Problem formulation and previous research

The idea of decomposition of a complex system into an organized set of simpler subsystems to simplify the system description is not new. In cognitive science it is known as **Functional Analysis** [13]. The basic idea is that the system is viewed as computing a function. Functional analysis is a process of decomposing that function into a structure of simpler subfunctions in a hope that the result will be easier to explain (Occam razor principle). Each subfunction can be viewed as definition of certain concept[1]. Another important notion of cognitive science is **Functional Architecture** and it can probably be best defined by the following sentence [60]: *Specifying the functional architecture of a system is like providing a manual that defines some programming language. Indeed, defining a programming language is equivalent to specifying the functional architecture of a virtual machine.* The functional architecture contains a set of primitive operations or functions (concept definitions). This means that these basic functions cannot be explained by being further decomposed into less complex subfunctions and constitute the vocabulary of our programming language.

Using learning systems terminology specifying functional architecture (programming language) is equivalent to specifying classifier model while developing hypothesis corresponds to writing a program in this language.

There are two main approaches to the analysis of complex systems: **probabilistic** and **deterministic**. Probabilistic approach requires the existence of global probability distribution over the variables of the system and the decomposition consists on determination of a set of simplest possible marginal probabilities. Deterministic approach requires specification of the global relation over the variables of the system and the decomposition consists on determination of a set of simplest possible relations describing the system.

Probabilistic system can in general be represented by a contingency table, two- or multi-dimensional, each cell of which contains the number of times a combination of variables corresponding to that cell has been observed. These numbers, called frequencies, normalized to the total number of observations are called relative frequencies or probabilities. In such table sum of probabilities always sums up to 1. Figure 1a shows an example of contingency table with frequencies corresponding to every combination of the system variables.

Figure 1b was created from Figure 1a by setting up a threshold on the cell frequencies. All the cells with frequencies greater or equal to 70 were assigned the value 1, those with frequencies smaller than 70 were assigned the value 0. If we designate variables c and d to be the output variables then functions $c = f_1(a, b)$ and $d = f_2(a, b)$ are binary functions AND and OR respectively. If the threshold on the cell frequencies is set up on 50 instead of 70, we obtain the contingency table in Figure 1c. The Figure 1c doesn't represent a function anymore. It can be either interpreted as a function with noise or directed relation. Karnaugh maps for functions (relations) c and d corresponding to the contingency tables in Figures 1b and c are shown in Figures 1d and e.

The construction presented above shows the relation between probabilistic and deterministic approaches to the system analysis. Deterministic system can be viewed as a YES, NO simplification of the probabilistic system and often constitutes the only available information on the system. In many situations it may be impossible or unreasonable to collect the global frequency (probability) information which is statistically reliable but it is relatively easy to collect meaningful information on the relation between different variables of the system. Deterministic approach is also justified if many cells of the contingency table contain 0, or the cells contain only two distinct values (or values which are close to two distinct values) of frequency which may be replaced with 0s and 1s.

[1]Webster: Concept - an idea of what a thing in general should be

ab \ cd	00	01	11	10
00	77	57	3	2
01	1	110	12	1
11	12	28	200	1
10	0	423	21	52

a)

ab \ cd	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	1	0	0

b)

ab \ cd	00	01	11	10
00	1	1	0	0
01	0	1	0	0
11	0	0	1	0
10	0	1	0	1

c)

a \ b	0	1
0	00	01
1	01	11

cd

d)

a \ b	0	1
0	00,01	01
1	01,10	11

cd

e)

Figure 1: Contingency tables

In the presented research, decomposition theory and algorithms for both **deterministic** and **probabilistic** multiple-valued relations (systems) have been developed [29].

Constructive Induction (concept proposed first by Michalski [45]) is a two level learning method which consists of searching for new concepts in data first (search for an adequate representation space), and then searching for the best hypothesis within the space of new concepts. The process of discovering new concepts is equivalent to specifying a vocabulary (variables) to be used while writing a program (hypothesis) using certain programming language (functional architecture).

Constructive induction is based on a number of ideas and assumptions [4]:

- It is based on the idea the quality of the knowledge representation space is the most important factor in concept learning. If the representation space is of high quality (i.e. chosen attributes or descriptive terms are of high relevance to the problem at hand), learning process will be relatively easy and will likely produce hypotheses with high predictive accuracy. If the quality of the representation space is low (i.e. attributes are of little relevance to the problem), a learning process will be complex and no method may be able to produce good hypotheses.
- It searches for patterns in data and/or learned hypotheses, and uses them for proposing knowledge space transformations (that may expand or/and contract the space).
- It creates new descriptors (attributes or terms) that may be very complex, multilevel functions or transformations of the original descriptors).
- It postulates that produced concept descriptions should be comprehensible to human experts, so that they are relatively easy to interpret and express in terms and forms used by experts.

Many constructive induction methods have been developed and they are usually classified based on the strategy employed for generating new representation spaces. They may be classified into the following categories:

- **Data Driven Constructive Induction**
  The input data are analyzed and, based on the relationships between the input variables, changes are made in the representation space. [64], [6].
- **Hypothesis Driven Constructive Induction**
  Hypotheses generated in the second step are used for selection of the representation space. The whole process (both steps) is iterated until the satisfying solution is found [19], [48].
- **Knowledge Driven Constructive Induction**
  The new representation space is generated based on expert-provided domain knowledge [36], [37].
- **Multistrategy Constructive Induction** is a combination of any of the above types [36], [45].

The idea of using decomposition in order to construct a network of functional blocks (concepts) matching given data was presented first by Lendaris and Stanley [40], [39], [38]. They use the theory of decomposition of binary functions developed in [5], [14], and [33] as a tool for the development of self-organizing systems, networks of adaptive logic elements in particular. The structure of the network (hypothesis) is modified according to the constraints in the environment pertinent to the task (function to be learned known by the teacher). Structure of the network analyzed in [40] and [39] is a disjunctive cascade of universal logic elements similar to Maitra cascades [43]. Subfunctions relevant to the task are discovered in the process of adjusting parameters of universal logic elements until they match the learning data. The approach was applied to completely and incompletely specified binary functions.

The standard machine learning approach is focused on learning a single concept from data. In our approach various decompositions based on fundamental Ashenhurst-Curtis type decomposition are used to decompose data into an organized multilevel structure of primitive (nondecomposable) relations (concepts). All the relations may be both completely and incompletely specified, and variables can be shared between different relations.

The method can be classified as multistrategy constructive induction method. An example of application of the method is shown in Fig. 2.

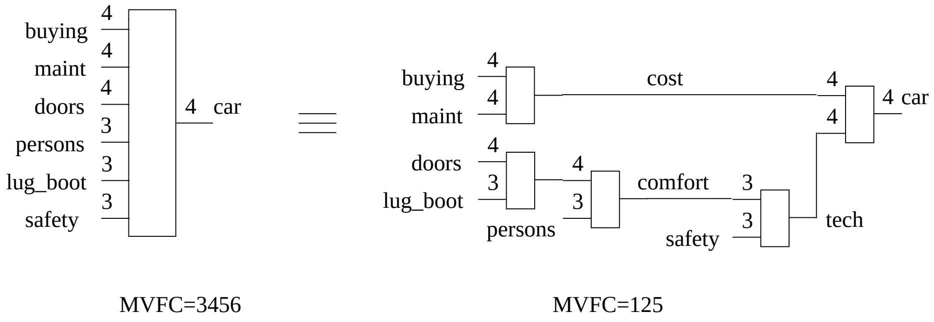


Figure 2: Discovering new concepts

The benchmark `car` described in [7] was developed for evaluating cars based on their price and technical characteristics. Three new concepts (variables) have been discovered in the decomposition process: *cost*, *comfort*, and *tech*. Concept *cost* depends on input variables *buying* (buying price), and *maint* (maintenance cost). Concept *comfort* depends on input variables *doors* (number of doors), *lug_boot* (luggage boot), and *persons* (number of persons). Variables *safety* and *comfort* define concept *tech* (technical characteristics), and the original concept *car* can be now expressed in terms of the concepts *tech* and *cost*. The complexity of the new structure is much smaller than the original one and according to the Occam Razor principle should have better generalization properties.

In our approach, data driven constructive induction is performed in the attribute reduction (vacuous variables removing) and variable partitioning steps while the hypothesis driven constructive induction is performed in the automatic hypothesis selection step. Possible extensions include knowledge-driven constructive induction by allowing the user to participate in the hypothesis selection process.

B. Decomposition of Multiple Valued functions and relations

Decomposition of Multiple-Valued functions emerged as an extension of approaches known from the binary functions domain. The theory of decomposition of binary functions [5], [14] was extended to multiple-valued, completely and incompletely specified functions by Karp [33]. The approach presented by Walliuzzaman [72] resembles Curtis approach for binary functions [14] but when Curtis investigates all possible decompositions to select the best one, Walliuzzaman develops a set of conditions for easy selection of decomposition that leads to simple implementation. Abugharrbieh and Lee [2] and [3] extend Shen's algorithm for binary functions [68] on the multiple-valued functions. Their method operates on functions that may be given either by truth tables or algebraic expressions. Luba [42] uses partition based method to decompose multiple-valued functions. All the above algorithms start from the initial function and decompose it step by step by extracting smaller subfunctions. Another, compositional approach to decomposition of multiple-valued functions is presented by Fang and Wojcik [20]. In their approach original function is expressed by a composition of subfunctions taken from a library of already predefined functions. Another approach presented in [18] is based on representation of multiple-valued functions in terms of MTMDD (multi-terminal, multiple-valued decision diagrams). MTMDD approaches however were used previously only for disjoint decomposition of **completely specified** functions. None of the above algorithms present decomposition results for functions with more then 10 input variables and they assume that functions are homogeneous (all the variables are of equal multiplicity). In addition, the MTMDD based methods were used for disjoint decomposition only.

In terms of constructive induction terminology, Ashenhurst-Curtis type decomposition corresponds to discovering new concepts from data while the approach presented by Fang, to describing data in terms of predefined, existing concepts.

Another interesting approach to decomposition of directed systems [2] based on reconstructability analysis was presented by Zwick [81]. He uses a structural model obtained from reconstructability analysis (neutral systems[3]) and uses the maximum uncertainty condition to the subsystems of the model as constraints to obtain the relationship between input and output variables. The network consists of two levels: level 1 containing new concepts discovered in reconctructability analysis and level 2 containing the hypothesis formulated in terms of the concepts at level 1. All the blocks of the structure can be relations, sets of inputs to different blocks can overlap, and the structure is always a two level structure.

Comprehensive review of existing methods, and presentation of new ideas for functional decomposition for binary, multiple-valued and continuous functions is presented in [49, 54, 52]. Based on that work new algorithms for functional decomposition of incompletely specified, multiple-valued functions were developed and implemented in program GUD [53]. GUD, part of Multis system, was the first program able to perform multi-level decomposition of large multiple-valued functions. Some of the ideas presented in [54], [52], and [53] were also implemented in programs HINT [77, 78, 79, 80] and Fred [23]. Theory and implementation of the first multi-level decomposer (MVGUD) for large multiple-valued directed relations were presented in [56]. Several new approaches to decompositions and partial problems on functions and relations were next developed using both labeled rough partitions [29], multi-valued decision diagrams [27] and implicit algorithms and representations [46, 47].

In the course of our work, driven by practical test cases, we strived for high processing speed and large size of data - thus we developed several logic function and relation manipulation packages for decomposition. They used **various data representations**: cube calculus [74, 73], partition calculus [42, 54], Binary Decision Diagrams [54], Cube Diagram Bundles [50], Multivalued Decision Diagrams [23], Binary-Coded Multi-valued Decision Diagrams [27], Labeled Rough Partitions [56, 28], and BDD-based implicit representation [46, 47]. As mentioned before, the choice of good representation is absolutely crucial to the overall success of the decomposer and is the single most important factor leading to eventual success.

We developed a general method of decomposition of neutral relations (systems), both deterministic and probabilistic [29, 30]. In this paper, however, we will present the decomposition process of deterministic directed relations only.

The method of decomposition of deterministic directed relations follows the main ideas from [56] but uses a different data structure to represent relations with noise. It transforms a multiple-valued incompletely specified function or relation into a multi-level structure and doesn't depend on particular assumptions about the nature of the blocks of which the structure is to be composed. The transformation process is based on Ashenhurst-Curtis type serial decomposition.

One step of Ashenhurst-Curtis type decomposition consists of forming a description of the initial relation $y(X)$ in terms of other, less complex relation $g(X_1)$ and input variables X_2:

$$y(X) = f(g(X_1), X_2)$$

where X_1, X_2 are sets of input variables and $X = X_1 \cup X_2$, sets X_1 and X_2 may overlap. If X_1 and X_2 overlap decomposition is called nondisjoint, otherwise it is called disjoint.

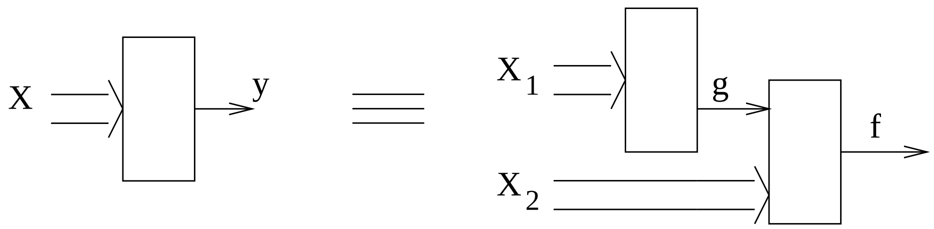


Figure 3: One level serial decomposition: $y(X) = f(g(X_1), X_2)$

Original relation $y(X)$ is now being represented in terms of variables of the new representation space $\{g, X_2\}$, more suitable for the problem description. The initial representation space $X = X_1 \cup X_2$ has been divided into two subspaces (not necessarily disjoint) and one of them used to define a new concept $g(X_1)$ (see Figure 3). The selection of X_1, X_2, and g is carried out as to minimize the overall complexity measure of the result. According to the general Occam razor principle this should result in better generalization properties of the selected hypothesis.

[2] directed system - system with specified inputs and outputs
[3] neutral system - system with no distinction between inputs and outputs

The decomposition process is repeated iteratively until terminating criteria is met. At each decomposition level the local optimization is performed and the resulting blocks are relations. Once the decomposition is terminated these relations provide extra choices for the final, global optimization.

The idea of Ashenhurst-Curtis type decomposition for multiple-valued relation is presented in Example B. For the purpose of this example Karnaugh map representation is used to make the presentation of the idea more clear. In the programs, however, decision diagrams, lr-partition or other abovementioned representations were used to reduce the data size and increase the decomposition speed.

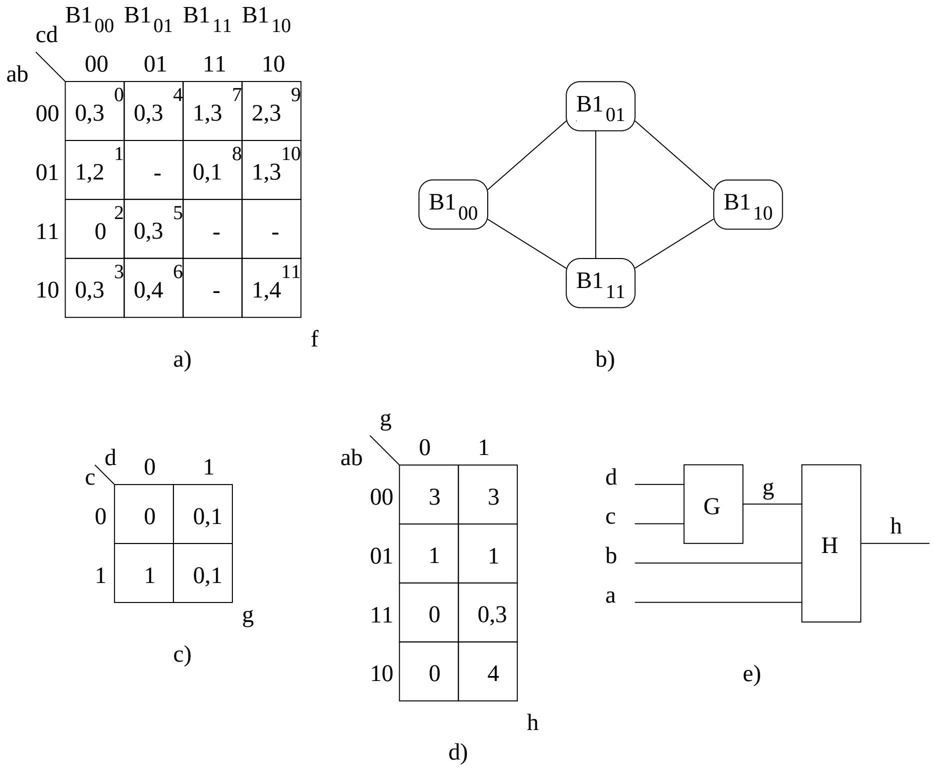


Figure 4: Decomposition of multiple-valued relation

Example A.

Given is a relation with 4 binary input variables and a 5-valued output variable represented by the Karnaugh map in Figure 4a. The bound set $X_1 = \{c, d\}$, free set $X_2 = \{a, b\}$, and output set $Y = \{f\}$. Each column of the Karnaugh map corresponds to a different combination of the bound set variables. In order to minimize complexity of the decomposition result for given X_1 and X_2 we want to minimize the number of compatibility classes the columns of the table are to be assigned to. A set of columns can be assigned to the same compatibility class (is a set of compatible columns) if, for every row of the table, there exists at least one value which is common for all the columns in the set. We can reduce the problem of compatibility classes determination to the graph covering problem. Column compatibility graph consists of vertices corresponding to the Karnaugh map columns, and edges, edge joins two vertices iff the corresponding columns are compatible. Column compatibility graph for the Karnaugh map in Figure 4a is shown in Figure 4b. The minimum number of cliques to cover the graph is 2. Let us select maximum cliques $C_0 = \{B1_{00}, B1_{01}, B1_{11}\}$ and $C_1 = \{B1_{01}, B1_{10}, B1_{11}\}$ to cover the graph. Each clique corresponds to a separate values of the output of block G (variable g in Figure 4e). Karnaugh maps for relations g and h are shown in Figures 4c and d respectively assuming clique C_0 corresponds to value 0 and clique C_1 corresponds to value 1 of the new variable g. Each compatibility class of columns is in Figure 4d reduced to the column of values all the columns of the class have in common. Since the cliques selected to cover the graph were nondisjoint, $g = G(X_1)$ is a relation. If cliques were disjoint (cliques $\{B1_{00}, B1_{01}, B1_{11}\}$ and $\{B_{10}\}$ for instance) $G(X_1)$ would be a function.

There are several partial problems that can be either solved efficiently, or totally avoided in some special decomposition variant. Column compatibility problem can be also reduced to graph coloring for which we found several efficient algorithms. We showed that although the problem is NP-hard, for decomposition data of realistic sizes exact coloring is not needed and our algorithm gives practically the same quality decompositions as one using exact coloring, which however cannot be used for larger functions [44, 58]. Another problem important in binary decomposition is encoding of columns. We developed algorithms for encoding [55, 10, 27], as well as non-traditional way of using them in multi-valued decomposition for binary applications [27]. Paper [9] demonstrates how essential savings in time can be obtained when a new method of "table creation" is applied to large bound sets.

Table 2 shows the result of comparison of MVGUD decomposer [29] to leading binary decomposers in terms of cost of the final results. All the functions in the table are binary and are taken from the set of MCNC benchmarks. TRADE is a decomposer developed at Portland State University, MISII at University of California, Berkeley, and DSGN174 is a decomposer developed under supervision of Prof. Steinbach in Germany. The final cost value is computed as a sum of the costs of single blocks of the result of the decomposition. The cost of a single block is computed using equations 2. For our program (mvgud) there is also execution time given (DECstation 5000/240, 64 MB of memory, user time in seconds) to show that the decomposition task can be performed in a reasonable amount of time.

		cost				
File	i/o	TRADE	MISII	DSGN174	mvgud	[time]
5xp1	7/10	496	384	292	236	[11.0]
9sym	9/1	640	984	400	104	[26.4]
con1	7/2	80	68	60	70	[2.3]
duke2	22/29	6516	2428	2200	2896	[11289.0]
ex5p	8/63	-	3720	1560	2104	[208.0]
f51m	8/8	372	392	240	177	[10.1]
misex1	8/7	472	208	224	229	[8.6]
misex2	25/18	548	464	436	392	[1086.0]
misex3	14/14	9816	4204	3028	1744	[1316.0]
rd53	5/3	120	96	84	60	[1.8]
rd73	7/3	320	352	256	113	[13.1]
rd84	8/4	508	672	320	171	[32.6]
sao2	10/4	1848	516	468	441	[47.2]

Table 2: Decomposition of binary (MCNC) benchmarks

The underlined results are the best for a given benchmark. In 70% of cases MVGUD decomposer yielded the best results. Comparisons of our other decomposers results to themselves and to decomposers from literature can be found in [29, 26, 27].

IV. An Example.

Our approach allows to decompose not only binary but also multiple-valued functions. An example of such a function is the well known in machine learning community benchmark `trains`. The problem was proposed more than 20 years ago by Ryszard Michalski [36]. There are 10 trains (Figure 5), five going East, five going West, and the problem is to find the simplest rule which, for a given train, would determine whether it is East or Westbound. The best rules discovered at that time were:

1. If a train has a short closed car, then it Eastbound and otherwise Westbound.
2. If a train has two cars, or has a car with a jagged roof then it is Westbound and otherwise Eastbound.

Our programs use data representations derived from well known from logic synthesis domain Espresso format. For instance, the original problem description was transformed to MVGUD format, which resulted in the following data file:

```
.type mv
.i 32
.o 1
.ilb size load w0 l0 s0 n0 ls0 w1 l1 s1 n1 ls1 w2 l2 s2 n2 ls2 w3 l3 s3 n3 ls3
     a b c d e f g h i j
.ob direction
.imv 3 4 2 2 10 4 4 2 2 10 3 4 2 2 7 3 4 2 2 8 2 3 2 2 2 2 2 2 2 2 2 2
.omv 2
2 3 0 1 6 3 2 0 0 8 1 3 1 1 6 1 1 0 0 6 1 0 0 1 0 0 0 1 0 0 1 0  0
1 2 0 0 9 1 3 0 0 7 1 2 0 0 0 2 0 - - - - - 0 1 0 1 0 0 0 0 0 0  0
1 1 0 0 6 1 0 0 0 4 1 3 1 1 0 1 3 - - - - - 0 0 0 0 1 0 1 0 0 0  0
```

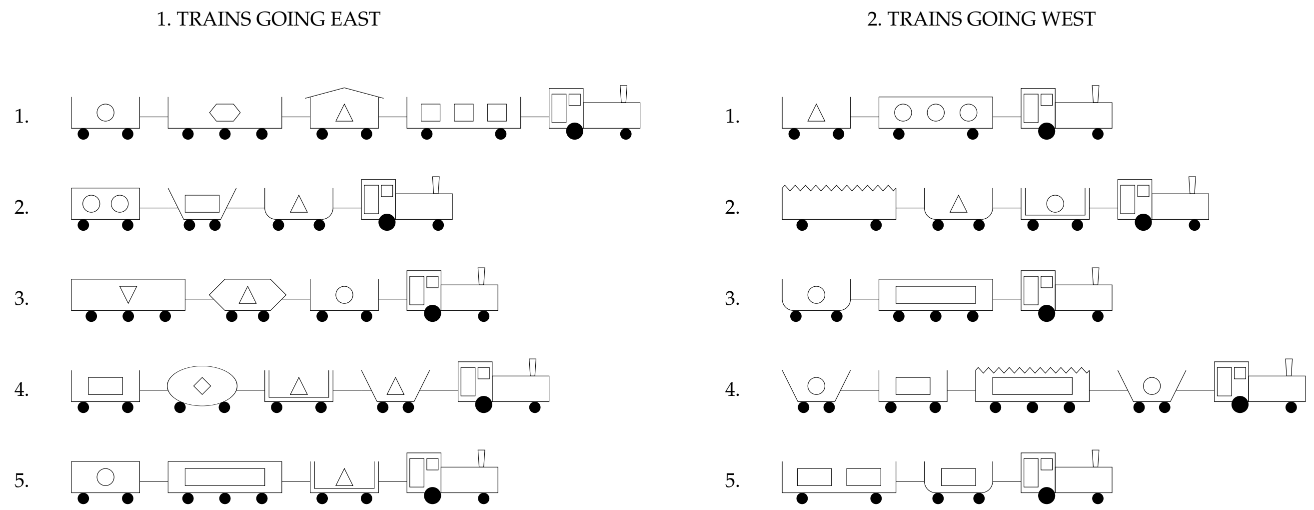


Figure 5: Trains problem

```
2 1 0 0 7 1 3 0 0 1 1 3 0 0 2 1 2 0 0 6 1 2 1 1 0 0 1 0 0 0 0 0  0
1 2 0 0 1 1 3 1 1 0 1 2 0 0 0 1 0 - - - - - 0 1 0 1 0 0 0 0 0 0  0
0 1 0 1 0 3 0 0 0 6 1 3 - - - - - - - - - - 0 0 0 0 0 0 1 0 0 0  1
1 1 0 0 1 1 0 0 0 9 1 3 0 1 5 0 - - - - - - 0 0 0 0 0 0 1 0 0 0  1
0 1 1 1 0 1 2 0 0 9 1 0 - - - - - - - - - - 0 0 0 1 0 0 0 0 0 0  1
2 1 0 0 7 1 0 0 1 5 1 2 0 0 6 1 2 0 0 7 1 0 1 0 0 1 0 0 0 0 0 0  1
0 0 0 0 9 1 2 0 1 6 2 2 - - - - - - - - - - 1 0 0 0 0 0 0 0 0 0  1
.end
```

where:

`.i`	number of input variables (attributes)
`.o`	number of output variables (attributes)
`.ilb`	input variable names
`.ob`	output variable names
`.imv`	cardinalities of input variables
`.omv`	cardinalities of output variables

Variables 1-2: general attributes

- `size` number of cars (integer in [3-5])
- `load` number of different loads (integer in [1-4])

Variables 3-22: 5 attributes for each of cars 2 through 5: (20 attributes total)

- `w` number of wheels (integer in [2-3])
- `l` length (short or long)
- `s` shape (closedrect,dblopnrect,ellipse,engine,hexagon, jaggedtop, openrect, opentrap, slopetop, ushaped)
- `n` number of loads (integer in [0-3])
- `ls` load shape (circlelod, hexagonlod, rectanglod, trianglod)

Variables 23-32: 10 Boolean attributes describing whether 2 types of loads are on adjacent cars of the train

- `a` rectangle next to rectangle (0 if false, 1 if true)
- `b` rectangle next to triangle (0 if false, 1 if true)
- `c` rectangle next to hexagon (0 if false, 1 if true)
- `d` rectangle next to circle (0 if false, 1 if true)
- `e` triangle next to triangle (0 if false, 1 if true)
- `f` triangle next to hexagon (0 if false, 1 if true)
- `g` triangle next to circle (0 if false, 1 if true)
- `h` hexagon next to hexagon (0 if false, 1 if true)
- `i` hexagon next to circle (0 if false, 1 if true)
- `j` circle next to circle (0 if false, 1 if true)

Attribute 33: Class attribute (east or west)
    `direction`   (east = 0, west = 1)
The number of cars vary between 3 and 5. Therefore, attributes referring to properties of cars that do not exist (such as the 5 attriubutes for the "5th" car when the train has fewer than 5 cars) are assigned a value of "-".

Applied to the **trains** problem our program discovered the following rules:

1. If a train has triangle next to triangle or rectangle next to triangle on adjacent cars then it is Eastbound and otherwise Westbound.

2. If the shape of car 1 (s1) is jagged top or open rectangle or u-shaped then it is Westbound and otherwise Eastbound.

As we can see the rules discovered using decomposition approach implemented in our program yielded solutions of complexity comparable to the best known solutions to this problem.

The results of decomposition of selected benchmarks from University of California Irvine machine learning repository are shown in Figures 6, 7, 8, 9. It is interesting to see how the new concepts have been created automatically by the program. The total costs of the resulting functions are greatly reduced comparing to the original ones. Also, in many cases, some input variables were found to be vacuous which additionally contributed to the final cost reduction. We developed several strategies and algorithms to deal with vacuous and similar types of variables that can be removed [29, 23, 27, 46].

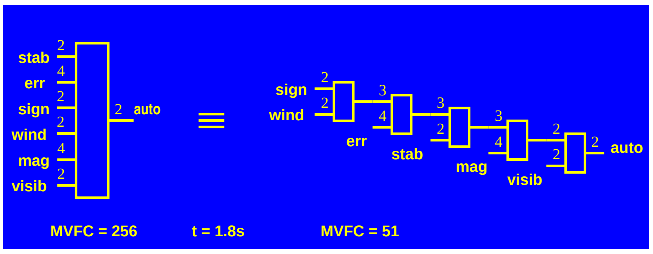


Figure 6: MV benchmarks: shuttle

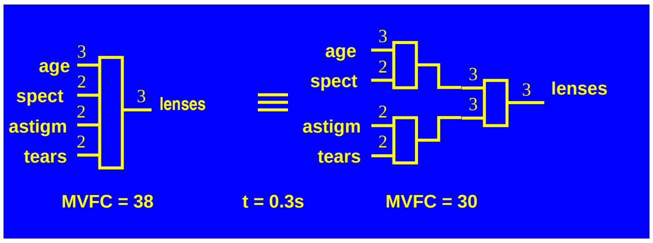


Figure 7: MV benchmarks: lenses

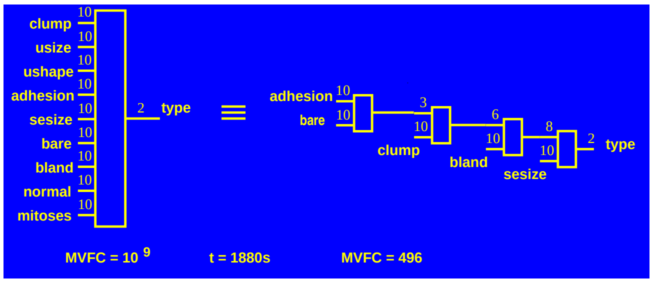


Figure 8: MV benchmarks: breastc

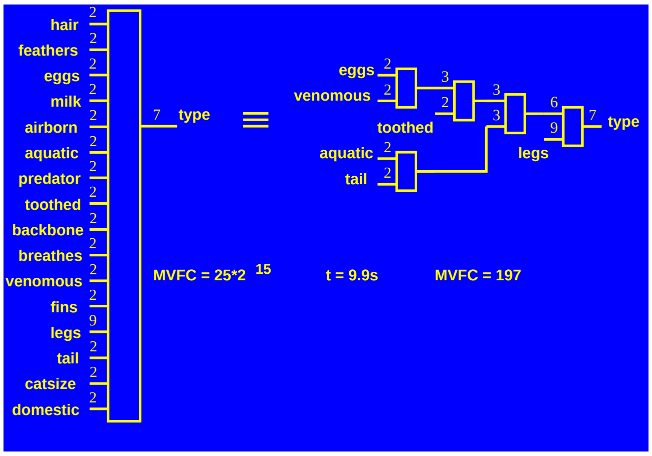


Figure 9: MV benchmarks: zoo

V. Variable partitioning for decomposition

Variable partitioning for decomposition is a process of splitting up the set of relation's variables X into subsets X_1 and X_2 in such a way that the decomposed relation is less complex than the initial one. Even then the problem of variable partitioning is very important not much has been done in this domain in the past. All efficient algorithms for this task have been created after 1994 [49, 50, 23, 47]. This is mainly because most of the earlier published decomposition algorithms were tested on small functions only and exhaustive search for partitions was possible in reasonable amount of time. In general the problem of optimal variable partitioning is NP-complete and fast heuristic procedures are needed to perform this task effectively. Two of such procedures were developed in [74] for binary functions, and were extended for multiple-valued relations and tested [29]. Another method, based on entropy measure, was developed in [29] to generate a limited set of pairs $\{X_1, X_2\}$ and select the one which would minimize the cost of the decomposed relation (concept). The method uses entropy [67] and variety [12] measures to order input variables according to their relevancy for the output variable determination (see Figure 10).

- **Uncertainty (Shannon):**

$$u(a) = -\sum_i p(a = a_i)\log_2 p(a = a_i)$$

- **Conditional Uncertainty (Shannon):**

$$u(a|b) = u(ab) - u(b)$$

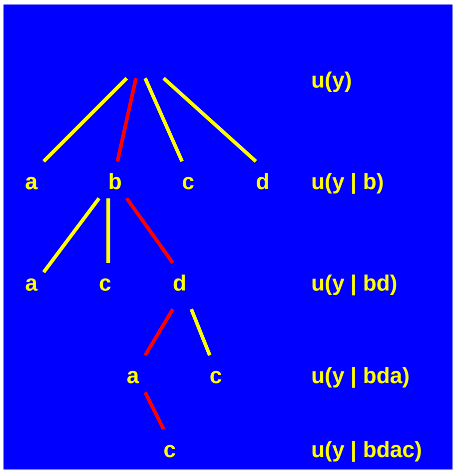


Figure 10: Variable ordering

Ordered set of input variables will be then partitioned into bound (X_1) and free (X_2) sets for decomposition. The optimal partitioning criteria will be determined based on the cost function used in the decomposition process. For the cost measures discussed in section A the selection of sets X_1 and X_2 is based on the following criteria:

The partition of a set of input variables X minimizing MVFC consists of a free set X_2 containing the most informative variables of the relation, and a bound set X_1 containing the remaining input variables where X_2 satisfies the following equation:

$$0.5\log_2 \prod_{x_i \in X} |x_i| = \log_2 \prod_{x_j \in X_2} |x_j|$$

where: $|x_i|$ is cardinality of variable $x_i \in X$,
x_j are the most informative variables of the relation

We observed that especially efficient partitioning methods can be developed when the number of variables in the bound set is a small [27] or large [9] percentage of all variables.

A. Complexity measure for multiple-valued relations

An appropriate complexity measure together with variable partitioning algorithms are of crucial importance for the quality of hypothesis selection process. Hypotheses are generated by different partitions X_1, X_2 of the input variables, the best is selected based on the cost function used. Complexity measure used as a starting point in this paper was the *normalized circuit complexity* proposed by Abu-Mostafa [1] for binary functions. He defined complexity C_x of a binary function $Y = f(X)$ as follows:

$$C_x(f) = \log_2 \min\{cost\ of\ \ \Gamma : \Gamma\ simulates\ f\} \tag{1}$$

where Γ is a combinational circuit realizing function f and cost is equal to 2^n for n-inputs universal block and the cost of a collection of blocks is the sum of the costs of the blocks.

Following his definition the cost of a single $|X|$-inputs $|Y|$-outputs universal block is equal to:

$$cost(f) = 2^{|X|}|Y| \tag{2}$$

where: $|X|, |Y|$ are cardinalities of sets of input and output variables respectively.

According to his definition the cost of a binary function realized by a single block is equal to the number of cells of the Karnaugh map representing the function which is equal to the total number of tuples defining the function. The larger that number is, the more variety the function can store, more details describe, and more difficult will be the physical realization of that function.

The cost of a single block is closely related to its Kolmogorov complexity. Every relation (function in particular) can be represented by a binary vector of length n, n equal to the total number of tuples representing the function or relation, and can be considered to be a program describing that relation. The length of the vector n can be interpreted as the length of the program and it bounds from above the value of Kolmogorov complexity $K(\cdot)$ for that relation.

The first definition of cost (normalized) used is based on Abu-Mostafa's definition but applies to multiple-valued variables as well.

$$C_x = \log_2 \left(\prod_{x_i \in X} |x_i| \ \log_2 \prod_{y_j \in Y} |y_j| \right) \tag{3}$$

where: $|x_i|$ is cardinality of variable $x_i \in X$,
$|y_j|$ is cardinality of variable $y_j \in Y$.

Justification for this formula is the following: let's define relation cost (denormalized) as the maximum number of tuples, x_i be an input variable, and y be $|y|$-valued output variable. Hence the maximum number of tuples (cost) is equal to the product $\prod_{x_i \in X} |x_i|$. If we add more outputs y_j, each will correspond to a separate function. If all of them are of equal cardinality m then the maximum number of tuples (cost) will be equal to $\prod_{x_i \in X} |x_i|$ times the number of outputs. If we allow y_j to have different cardinalities $|y_j|$ then the number of equivalent m-valued outputs will be equal to $\log_m \prod_{y_j \in Y} |y_j|$ and cost equal to $\prod_{x_i \in X} |x_i| \log_m \prod_{y_j \in Y} |y_j|$. If we choose $m = 2$ as our base variable cardinality then the logarithm will be base 2 and cost computed in respect to the number of equivalent binary outputs.

The above cost measure significantly reduced complexity of the final hypothesis. It is not the only possible cost measure however and analysis and testing of other solutions was used for performance evaluation.

Another definition was provided by Lendaris and Stanley [40] and it defines the cost as being equal to the total number of functions that can be realized by a given structure. For a single output binary function $y = F(X)$ the cost (denormalized) will be equal to:

$$C_x = 2^{2^{|X|}} \tag{4}$$

The normalization yields:

$$C_x = \log_2 \ 2^{2^{|X|}} = 2^{|X|} \tag{5}$$

Their formulae can be extended for multiple-valued, multi-output functions and directed relations as follows:

$$C_x = \log_2 \left(\prod_{y_j \in Y} |y_j| \right)^{\prod_{x_i \in X} |x_i|} = \prod_{x_i \in X} |x_i| \, \log_2 \prod_{y_j \in Y} |y_j| \tag{6}$$

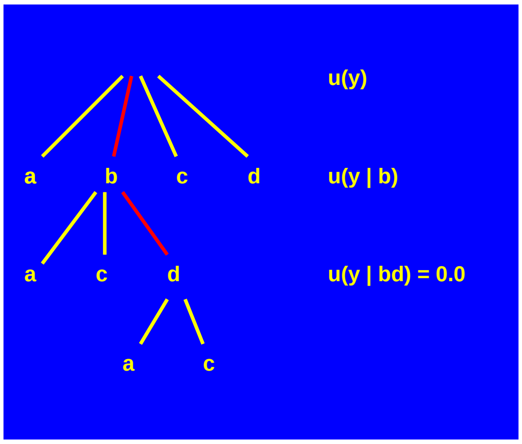
The above cost measure was also tested and compared to the previous one. Some other cost measures have been analyzed, tested, and compared by us to provide a broader picture of the problem of cost measures for various decomposition types, [23, 24, 25, 26, 27, 31, 29, 30].

B. Data and dimensionality reduction

Real life data are often redundant, containing many vacuous variables (having no significant impact, if at all, on the classification result), which may obscure, otherwise obvious, relations and dependencies. Vacuous variables (attributes) and data samples may also significantly increase time complexity of the learning algorithms and lead to more complex solutions. The data and dimensionality reduction becomes though an important step of the learning algorithm and may significantly decrease not only the time complexity but also leads to simpler solutions, with better predictive accuracy.

In our approach several different methods of reduction of vacuous variables are proposed, one incorporated into the variable partitioning process and the other, based on different principles, are included in the decomposition algorithm. The reduction of redundant data is performed at each level of multi-level decomposition process.

For instance, one of the methods, based on the conditional uncertainty, was incorporated into variable partitioning procedure (see Section V). The procedure of variable ordering computes conditional uncertainties of input variables in respect to the output variables. If, after investigating a successive variable, uncertainty reduces to zero it implies that all the remaining variables are irrelevant for the output variables determination and can be eliminated from further investigation (Figure 11).



- Variables b and d reduce uncertainty of y to 0 which means they provide all the information necessary for determination of the output y
- Variables a and c are vacuous

Figure 11: Vacuous variables removing

The second method can be applied at each step of decomposition procedure. If in a given decomposition step (Figure 3) function $g(X_1)$ is a constant function then the function $h(g(X_1), X_2)$ doesn't depend of variables $x_i \in X_1$. Hence, all the variables $x_i \in X_1$ are vacuous and can be removed from further analysis.

VI. Dealing with noise

Real life data sets used for building classifiers are hardly ever perfect and have often incorrect or uncertain values of variables (attributes). The way to deal with the problem usually depends on whether it is an input or output variable.

The most common types of errors are:

- Missing value.
  The most common technique to handle missing value is to either neglect the whole data sample or substitute the value with a value selected according to certain criteria, for instance select the value the variable takes most often in the same class. If the value of the output variable is unknown the whole data sample is neglected.

- Incorrect value.
  The incorrect values result from inaccurate measurements or reading errors. The difference between correct and actual value is commonly referred to as noise. Different techniques can be used to handle the problem and quality often depends on the data availability. The more data we have available the better the result is. Incorrect values can be adjusted in the preprocessing stage (noise removal) or the learning system alone takes them into account and generates classifier which maintains a high level of accuracy in the presence of noise.

- Uncertain value.
  In some situations it is more appropriate, or even necessary, to state that a variable can take few out of a set of possible values instead of limiting it to a single value. For instance, 'if the color of the street light is green or yellow you can pass', or 'only small or medium size packages are accepted'. The same applies to the output variables. Their values, in the case of supervised learning, have to be determined by a teacher, expert in the area (class assignment). However, since experts not always agree on what class a given vector of input values has to be assigned to, it is desirable to postpone the decision until additional circumstances allow us to make a justified decision. The decision can be postponed until the learning process is finished and then the right value can be selected to minimize overall complexity of the classifier. This however, requires representing a problem in terms of a relation instead of function, and learning method handling more general notion of relation as well.

The presence of noise in the data adds a new dimension to the problem of selection of hypothesis. First of all, the procedure of construction of compatibility graph (Example B) has to be modified to address overfitting problem (hypothesis which perfectly fits the data with noise has usually poor generalization properties).

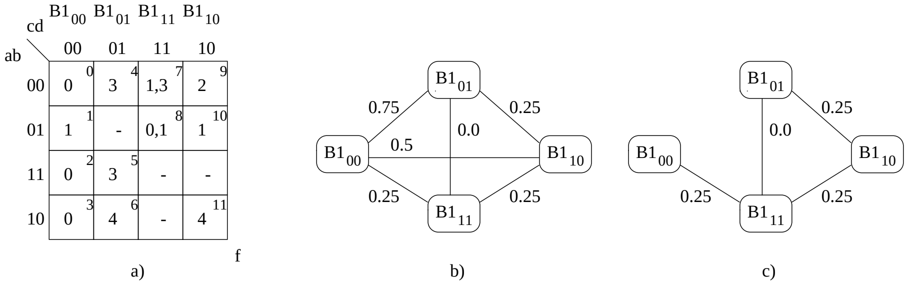


Figure 12: Compatibility graph construction for data with noise

The graph in Figure 12b is constructed for data represented by the Karnaugh map in Figure 12a. It is a full graph with weight assigned to every edge. Each weight takes values between 0.0 and 1.0 and is equal to the minimum number of mismatches between the columns connected by that edge, normalized to the maximum possible number of mismatches (number of rows). YES or NO selection criterion removes all the edges from the graph which weights are greater or equal to certain value d. Figure 12c shows the graph obtained for $d = 0.5$. A procedure which would directly use the full information contained in the graph in Figure 12b can also be developed.

Second, in the presence of noise complexity measures discussed earlier are not sufficient anymore. In order to select the best hypothesis both the complexity and error (losses) have to be taken into account (different d result in different complexities and errors).

To address this problem a minimum description length principle combining complexity measure and error evaluation procedure has been developed and incorporated into hypothesis selection process.

The algorithms developed in the previous sections learn in nominal data spaces. It means that the values of variables are mere symbols and the data space is not metric (no distance measure can be defined on it). This is the most general approach for discrete data spaces. The data in the ovulation databases however, are continuous, they have to be discretized before being used by our procedures, and the resulting data space is metric. Hence, two issues need to be analyzed here. First of all, the use of appropriate discretization method may increase not only the accuracy but also significantly increase the learning speed [17], [11]. The most often used discretization method, uniform binning, divides the space of each variable values into a number of equally sized bins. Another type of methods are based on the entropy measure [11], [22] and use minimum entropy criterion to assign the values to different bins. These methods result in better performance of learning systems than the uniform binning methods [17].

The second, the algorithms developed for the nominal data spaces can be modified to take into account metricity of the data space. This may again significantly decrease a complexity of the results. Example is given below.

Figure 13 shows the process of constructing compatibility graph in two cases: nominal data and metric data. For nominal data no distance measure can be defined which implies for instance that we can not say that 1 is

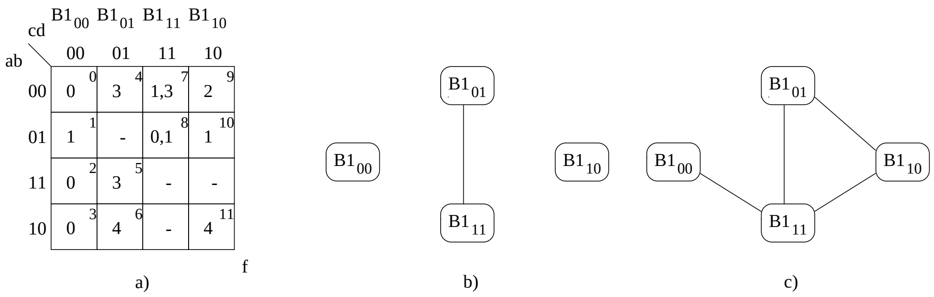


Figure 13: Compatibility graph for metric data

closer to 2 than to 5. These are all mere symbols, not numbers. For metric data however, number 1 is closer to 2 than to 5 and that fact can be used to consider some columns to be compatible even though the values in corresponding rows are not equal. Graph in Figure 13b results from interpreting data in the table in Figure 13a as being nominal. Two columns are compatible only if they have a common value in every row of the table. Graph in Figure 13c was created when interpreting the data as being metric. In this case columns are considered to be compatible if the difference (distance) between values of the corresponding row cells are not greater than 1. The graph constructed according to this assumption can be covered by fewer cliques (2 vs. 3) and results in simpler blocks G and H. Other distance measures can be used as well. This method works also well with noisy data and prevents development of overfitted hypotheses.

VII. Research results evaluation

Classifiers developed in the learning processes are evaluated based on the classification error rate:

$$error = \frac{\#\ of\ incorrectly\ classified\ samples}{total\ \#\ of\ samples}$$

The most common learning/evaluation procedure is to partition the set of available samples into two subsets: learning set and testing set. Learning is performed using data contained in the learning set and error rate computed by classifying data in the testing set. The calculated error rate is of course an estimation of the true error rate over the full sample space. The accuracy of estimation depends on the number of test cases available. For the test samples of 1000 or more the estimate is very close to the true error rate [76] and the method provides an accurate way for evaluating classifiers.

When the number of available samples is small other methods provide better true error rate estimation. The best results can be obtained using resampling techniques such as cross-validation (leave one out, k-fold cross-validation) and bootstrapping [76].

Evaluations of the learning processes and discussion can be found in [29, 27].

VIII. Conclusion

Stimulated by practical hard problems in logic synthesis using the technology of Field Programmable Gate Arrays (FPGA), logic design of Application Specific Integrated Circuits (ASIC) and high performance custom Very Large Scale of Integration (VLSI) processors, robotics, Machine Learning and Data Mining, we developed over the years a set of tools for decomposition of binary and multivalued functions and relations. The methods have been also extended to fuzzy logic [8], reconstructability analysis [29, 30] and real-valued functions [65, 16]. Our recent software allows also for bi-decomposition [69, 70, 47], removal of vacuous variables [46] and other preprocessing/postprocessing operations [11, 17]. Variants of our software are used in several commercial companies where they found various types of applications. Current applications include: epidemiology, hexapod robot gait control [41], FPGA synthesis [73, 35, 66] data mining medical databases [29, 30], and VLSI layout driven logic synthesis [27], but in theory the applications of the method are unlimited and it can be used whenever decision trees or artificial neural nets are used now. The quality of learning was better than in the top decision tree creating program C4.5 and various neural nets [27, 29]. The only problem that remains is speed in some applications.

On our WWW page, $http://www.ee.pdx.edu/\tilde{c}files/papers.html$ the reader can find many benchmarks from various disciplines that can be used for comparison of machine learning and logic synthesis programs. We plan to continue work on decomposition and its various practical applications such as epidemiology or robotics which generate large real-life benchmarks.

All in all, despite success of our approach, we consider the problem of functional decomposition as far from being solved from the practical standpoint. We believe that new decomposition types, new deep theoretical results, use of implicit methods, new relation/function representations, and use of special-purpose processors [57, 59, 32] and parallel processors will ultimately lead to even better decomposers in future.

References

[1] Y.S. Abu-Mostafa, *Complexity in Information Theory.* Springer-Verlag, New York, 1988.

[2] S.B. Abugharbieh and S.C. Lee, "A fast algorithm for disjunctive decomposition of m-valued functions. Part I: The decomposition algorithm," In *Proc. 23th ISMVL*, pp. 118–125, 1993.

[3] S.B. Abugharbieh and S.C. Lee, "A fast algorithm for disjunctive decomposition of m-valued functions. Part II: Time complexity analysis," In *Proc. 23th ISMVL*, pp. 126–131, 1993.

[4] T. Arciszewski, R.S. Michalski, and J. Wnek, "Constructive induction: the key to design creativity," Reports on machine learning and inference laboratory, mli 95-6, *George Mason University,* April 1995.

[5] R. L. Ashenhurst, "The decomposition of switching functions," In *Proc. Int. Symp. Theory of Switching, Part I*, pp. 74–116, Ann. Comput. Lab. Harvard Univ., 1959.

[6] E. Bloedorn and R.S. Michalski, "Data driven constructive induction in AQ17-PRE: A method and experiments," In *Proceedings of the Third International Conference on Tools for AI*, San Jose, CA, 1991.

[7] M. Bohanec and V. Rajkovic, "Knowledge acquisition and explanation for multi-attribute decision making," In *Proc. of the 8th Int. Workshop on Expert Systems and their Applications*, pp. 59–78, Avignon, France, 1988.

[8] P. Burkey and M. Perkowski, "Decomposition of fuzzy functions and relations," Report PSU, 2001.

[9] M. Burns, M. Perkowski, and L. Jozwiak, "An Efficient Approach to Decomposition of Multi-Output Boolean Functions with Large Sets of Bound Variables," *Proc. 1998 Euromicro,* pp. 16-23, Vasteras, Sweden, August 25-27, 1998.

[10] M. Burns, M. Perkowski, L. Jozwiak, and S. Grygiel, "An Efficient and Effective Approach to Column-Based Input/Output Encoding in Functional Decomposition," *Proc. 3rd Intern. Wrk. Boolean Problems,* pp. 19-29, Freiberg University of Mining and Technology, Sept. 17-18, 1998.

[11] J. Catlett, "On changing continuous attributes into ordered discrete attributes," In Y. Kodratoff, editor, *Proc. of the European Working Session on Learning*, pp. 164–178, Berlin, Germany, 1991. Springer-Verlag.

[12] R.C. Conant, "Set-theoretic structure modeling," *International Journal of General Systems*, 7(38):93–107, 1981.

[13] R. Cummings, *The nature of psychological explanation*, MIT Press, Cambridge, MA, 1983.

[14] H. A. Curtis. *A New Approach to the Design of Switching Circuits.* Van Nostrand, Princeton, 1962.

[15] M. DesJardins and D. F. Gordon, "Evaluation and selection of biases in machine learning," *Machine Learning Journal*, 20:5–21, 95.

[16] J. Demsar, B. Zupan, M. Bohanec, I. Bratko, "Constructing Intermediate Concepts by Decomposition of Real Functions," ECML 1997, pp. 93-107.

[17] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," In *Machine Learning: Proc. of the 12th International Conference*, San Francisco, CA, 1995. Morgan Kaufmann.

[18] E.V. Dubrova, J.C. Muzio, and B. von Stengel, "Finding composition trees for multiple-valued functions," In *Proc. 27th ISMVL*, 1997.

[19] W. Emde, C.U. Habel, and C.R. Rollinger, "The discovery of the equator or concept driven learning," In *Proceedings of IJCAI-83*, pp. 455–458, Karlsruhe, Germany, 1983. Morgan Kaufmann.

[20] K.Y. Fang and A.S. Wojcik, "Modular decomposition of combinational multiple-valued circuits," *IEEE Transactions on Computers*, 37(10):1293–1301, 1988.

[21] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery: An overwiev," In Fayaad at al., editor, *Advances in Knowledge Discovery and Data Mining.* MIT Press, 1996.

[22] U.M. Fayyad, P. Smyth, N. Weir, and S. Djorgovski, "Automated analysis and exploration of image databases: Results, progress, and challenges," *Journal of Intelligent Information Systems*, (4):1–19, 1993.

[23] C. Files, R. Drechsler, and M. Perkowski, "Functional decomposition of mvl functions using multi-valued decision diagrams," In *Proc. of ISMVL'97*, pp. 27–32, Halifax, Nova Scotia, Canada, May 28-30 1997.

[24] C. Files, and M. Perkowski, "Multi-Valued Functional Decomposition as a Machine Learning Method," Proc. ISMVL'98, May 1998.

[25] C. Files, and M. Perkowski, "An Error Reducing Approach to Machine Learning Using Multi-Valued Functional Decomposition," Proc. ISMVL'98, May 1998.

[26] C. Files and M. Perkowski, "New Multivalued Functional Decomposition Algorithm Based on MDDs," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems,* Vol. 19, No. 9, September 2000, pp. 1081 - 1086.

[27] C. Files, "A New Functional Decomposition Method As Applied to Machine Learning and VLSI Layout," Ph.D. Dissertation, Portland State University, June 2000.

[28] S. Grygiel, M. Perkowski, M. Marek-Sadowska, T. Luba, and L. Jozwiak, "Cube Diagram Bundles, A New Representation of Strongly Unspecified Multiple-Valued Functions and Relations," *Proc. ISMVL'97,* Halifax, Nova Scotia, Canada, May 1997, pp. 287 - 292.

[29] S. Grygiel, *Decomposition of relations as a new approach to constructive induction in machine learning and data mining*, Ph.D. thesis, Portland State University, 2000.

[30] S. Grygiel, M. Zwick, and M. Perkowski, "Decomposition in Reconstructability Analysis," Paper in preparation.

[31] S. Grygiel and M. Perkowski, "Labeled rough partitions - a new general purpose representation for multiple-valued functions and relations," *Journal of Systems Architecture,* Elsevier, No. 47, pp. 29 - 59, 2001.

[32] R. Hatt, "Design and Evaluation of a Specialized Computer Architecture for Manipulating Binary Decision Diagrams," *M.S. Thesis,* October 2000, Dept. ECE, PSU.

[33] R.M. Karp, "Functional decomposition and switching circuit design," *SIAM Journal on Applied Mathematics*, 11(2):291–335, June 1963.

[34] I. Kononenko, "Inductive and bayesian learning in medical diagnosis," *Applied Artificial Intelligence*, (7):317–337, 1993.

[35] Y. T. Lai, K.R. Pan, M. Pedram, and S. Vrudhula, "FGMap: A technology mapping algorithm for look-up table type FPGA synthesis," In *Proc. 30-th DAC*, pp. 642–647, 1993.

[36] J.B. Larson and R.S. Michalski, "Inductive inference of VL decision rules," *ACM SIGART Newsletter*, (63):38–44, 1977.

[37] D.B. Lenat, "On automated scientific theory formation: A case study using AM program," In *Machine Intelligence*, volume 9. Halsted Press, New York, 1977.

[38] G.G. Lendaris and G.L. Stanley, "On the structure-dependant properties of adaptive logic networks," Technical report, GM Defense Research Laboratories, Santa Barbara, California, July 1963.

[39] G.G. Lendaris and G.L. Stanley, "Self-oranization: meaning and means," In J. Spiegel and D. Walker, editors, *Proceedings of the Second Congress*, Information System Sciences. Spartan Books, Baltimore, 1965.

[40] G.G. Lendaris and G.L. Stanley, "Structure and constraints in discrete adaptive networks," In *National Electronics Conference*, volume XXI, 1965.

[41] M. Levy, and M. Perkowski, "Gait generation for a hexapod robot using functional decomposition". PSU Report, February 2001.

[42] T. Luba, "Decomposition of multiple-valued functions," In *Proc. 25th ISMVL*, pp. 256–261, 1995.

[43] K.K. Maitra, "Cascaded switching networks of two-input flexible cells," *IRE Transactions on Electronics Computers*, April 1962.

[44] R. Malvi, M. Perkowski, and L. Jozwiak, "Exact Graph Coloring for Functional Decomposition: Do we Need it?," pp. 1-10, *Proc. 3rd Intern. Workshop on Boolean Problems,* Freiberg University of Mining and Technology, Institute of Computer Science, September 17-18, 1998.

[45] R.S. Michalski, "Pattern recognition as knowledge-guided computer induction," Technical report no. 927, Department of Computer Science, University of Illinois, Urbana-Champaign, 1978.

[46] A. Mishchenko, C. Files, M. Perkowski, B. Steinbach, Ch. Dorotska, "Implicit Algorithms for Multi-Valued Input Support Manipulation," *Proc. of 4th Intl. Workshop on Boolean Problems,* September 2000, Freiberg, Germany.

[47] A. Mishchenko, B. Steinbach, M. Perkowski, "An Algorithm for Bi-Decomposition of Logic Functions," Submitted to *DAC 2001,* June 18-22, Las Vegas.

[48] K. Morik, "Sloppy modeling," In K. Morik, editor, *Knowledge Representation and Organization in Machine Learning*. Springer-Verlag, Berlin Heidelberg, 1989.

[49] M. Perkowski, "A Survey of Research Areas in Functional Decomposition," Avionics Directoriate, *Wrights Laboratories, Wright-Patterson Air Force Base,* Dayton, Ohio, 31 August 1994.

[50] M.A. Perkowski, "A New Representation of Strongly Unspecified Switching Functions and Its Application to Multi-Level AND/OR/EXOR Synthesis," *Proc. of the Second Workshop on Applications of Reed-Muller Expansion in Circuit Design,* Chiba City, Japan, 27-29 August 1995, pp. 143-151.

[51] M. Perkowski, T. Ross, D. Gadd, J. A. Goldman, and N. Song. Application of ESOP minimization in machine learning and knowledge discovery. In *Proc. Reed-Muller'95 Workshop*, pp 102–109, Chiba, Japan, August 1995.

[52] M.A. Perkowski, S. Grygiel, and the Functional Decomposition Group, Department of Electrical Engineering, "A Survey of Literature on Function Decomposition," Version IV, PSU Electrical Engineering Department Report, November 20, 1995.

[53] M. Perkowski, M. Burns, T. Luba, S. Grygiel, C. Stanley, R. Price, Z. Wang, J. Lu, P. Burkey, D. Manoharan, and S. Mohammad. Development of search strategies for MULTIS. Technical report, Portland State University, Electrical Engineering Department, Portland, OR, December 1995.

[54] M. Perkowski, T. Luba, S. Grygiel, P. Burkey, M. Burns, N. Iliev, M. Kolsteren, R. Lisanke, R. Malvi, Z. Wang, H. Wu, F. Yang, S. Zhou, and J.S. Zhang. Unified approach to functional decompositions of switching functions. Technical report, Portland State University, Electrical Engineering Department, Portland, OR, June 1995.

[55] M. Perkowski, M. Burns, R. Almeria, and N. Iliev. "Approaches to the Input-Output Encoding Problem in Boolean Decomposition," PSU Electrical Engineering Department Report, January 9, 1996.

[56] M. Perkowski, M. Marek-Sadowska, L. Jozwiak, T. Luba, S. Grygiel, M. Nowicka, R. Malvi, Z. Wang, and J. S. Zhang, "Decomposition of multiple-valued relations," In *Proc. of ISMVL'97*, pp. 13–18, Halifax, Nova Scotia, Canada, May 28-30 1997.

[57] M. Perkowski, S. Grygiel, Q. Chen, and D. Mattson, "Constructive Induction Machines for Data Mining," *Proc. Conference on Intelligent Electronics,* Sendai, Japan, 14-19 March, 1999.

[58] M. Perkowski, R. Malvi, S. Grygiel, M. Burns, and A. Mishchenko, "Graph Coloring Algorithms for Fast Evaluation of Curtis Decompositions," *The 36th ACM/IEEE Design Automation Conference (DAC 99),* New Orleans, LA, USA, June 21-25, 1999, pp. 125-130.

[59] M. A. Perkowski, A. N. Chebotarev, A. A. Mishchenko, "Evolvable Hardware or Learning Hardware? Induction of State Machines from Temporal Logic Constraints," *The First NASA/DOD Workshop on Evolvable Hardware (NASA/DOD-EH 99).* Jet Propulsion Laboratory, Pasadena, California, USA, July 19-21, 1999, pp. 129-138.

[60] Z.W. Pylyshyn, *Computation and cognition*. MIT Press, Cambridge, MA, 1984.

[61] J. R. Quinlan, "Induction of decision trees," *Machine Learning,* (1):81–106, 1986.

[62] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, Los Altos, California, 1992.

[63] J.R. Quinlan, "Comparing connectionist and symbolic learning methods," In S.J. Hanson, G.A. Drastal, and R.L. Rivest, editors, *Volume I: Constraints and Prospects*, Computational Learning Theory, chapter 15, pp. 445–456. MIT Press, 1994.

[64] L. Rendell, "Substantial constructive induction using layered information compression: Tractable feature formation in search," In *Proc. of IJCAI-85*, pp. 650–658, 1985.

[65] T.D. Ross, M.J. Noviskey, M.L. Axtell, J.A. Goldman, and D.A. Gadd, "On the decomposition of real-valued functions," In the *Third Intern Workshop on Post-Binary VLSI Systems*, Boston, May 1994.

[66] T. Sasao, "FPGA design by generalized functional decomposition," In T. Sasao, editor, *Logic Synthesis and Optimization*, pp. 233–258. Kluwer Academic Publishers, 1993.

[67] C.E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. University of Illinois Press, 1975 (first published in 1949).

[68] V.Y. Shen, A. C. McKellar, and P. Weiner, "An fast algorithm for the disjunctive decomposition of switching functions," *IEEE Trans. on Comput.*, C-20(3):304–309, March 1971.

[69] B. Steinbach, M. Perkowski, and Ch. Lang, "Bi-Decomposition in Multi-Valued Logic for Data Mining," *Proc. ISMVL'99,* May, 1999.

[70] B. Steinbach, Ch. Lang, and M. Perkowski, "EXOR-Decomposition of Sets of Functions," *Proc. 4th Intern. Workshop on Applications of the Reed-Mller Expansion in Circuit Design (Reed-Mller 99),* University of Victoria, Victoria B.C., Canada, August 20-21, 1999.

[71] L.G. Valiant, "A theory of the learnable," *Comm. ACM*, (27):1134–1142, 1984.

[72] K.M. Walliuzzaman and Z.G. Vranesic, "On decomposition of multiple-valued switching functions," *Computer Journal*, 13:359–362, 1970.

[73] W. Wan and M. Perkowski, "A new approach to the decomposition of incompletely specified multi-output function based on graph coloring and local transformations and its application to FPGA mapping," In *Proc. Euro-DAC*, pp. 230–235, 1992.

[74] W. Wan, *A new approach to the decomposition of incompletely specified functions based on graph coloring and local transformation,* Master's thesis, Portland State University, May 1992.

[75] Y. Watanabe and R.K. Brayton, "Heuristic minimization of multiple-valued relations," *IEEE Transactions on CAD*, 12(10):1458–1472, October 1993.

[76] S. M. Weiss and C. A. Kulikowski, *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[77] B. Zupan and M. Bohanec, "Learning concept hierarchies from examples by function decomposition," Technical report, *J. Stefan Institute,* Ljubljana, 1996.

[78] B. Zupan, M. Bohanec, I. Bratko, B. Cestnik, "A Dataset Decomposition Approach to Data Mining and Machine Discovery, *KDD,* 1997, pp. 299-302.

[79] B. Zupan, M. Bohanec, J. Demsar, I. Bratko, "Feature Transformation by Function Decomposition," *IEEE Intelligent System,* 13(2), pp. 38-43, 1998.

[80] B. Zupan, M. Bohanec, J. Demsar, I. Bratko, "Learning by Discovering Concept Hierarchies," *Artificial Intelligence*, 109(1-2): pp. 211-242, 1999.

[81] M. Zwick, "Control uniqueness in reconstructability analysis," *International Journal of General Systems*, 23(2), 1995.