Experimental evaluation of three partition selection criteria for decision table decomposition

Blaž Zupan and Marko Bohanec

Department of Intelligent Systems
Jožef Stefan Institute, 1000 Ljubljana, Slovenia
e-mail: {Blaz.Zupan,Marko.Bohanec}@ijs.si

Abstract

Decision table decomposition is a machine learning approach that decomposes a given decision table into an equivalent hierarchy of decision tables. The approach aims to discover decision tables that are overall less complex than the initial one, potentially easier to interpret, and introduce new and meaningful intermediate concepts. Since an exhaustive search for an optimal hierarchy of decision tables is prohibitively complex, the decomposition uses a suboptimal iterative algorithm that requires the so-called partition selection criterion to decide among possible candidates for decomposition. This paper introduces two such criteria and experimentally compares their performance with the criteria originally used for the decomposition of Boolean functions. Two of these criteria are additionally used to assess the overall complexity of discovered decision tables. The experiments highlight the differences between the criteria, but also show that in all three cases the decomposition may discover meaningful intermediate concepts and relatively compact decision tables.

Keywords

decision table decomposition, partition selection criteria, knowledge discovery

1 Introduction

A decision table provides a simple means for concept representation. It represents a dataset with labeled instances, each relating a set of attribute values to a class (output concept). Decision table *decomposition* is a method based on the "divide and conquer" approach: given a decision table, it decomposes it to a hierarchy of decision tables. The method aims to construct the hierarchy so that the new decision tables are less complex and easier to interpret than the original decision table.

The decision table decomposition method is based on function decomposition, an approach originally developed for the design of digital circuits [2]. The method iteratively applies a *single decomposition step*, whose goal is to decompose a function $y = F(X)$ into $y = G(A, H(B))$, where X is a set of input attributes $x_1, \ldots, x_n$, and y is the class. F, G and H are functions represented as *decision tables*, i.e., possibly incomplete sets of attribute-value vectors with assigned classes. A and B are subsets of input attributes such that $A \cup B = X$. The functions G and H are developed in the decomposition process and are not predefined in any way. Such a decomposition also discovers a new intermediate concept $c = H(B)$. Since the decomposition can be applied recursively on H and G, the result in general is a *hierarchy* of decision tables.

Each single decomposition step aims to minimize the joint complexity of G and H and executes the decomposition only if this is lower than the complexity of F. Moreover, it is of crucial importance for the algorithm to find such partition of attributes X into sets A and B that yields G and H of the lowest complexity. The criteria that guide the selection of such partition are called *partition selection criteria.*

Let us illustrate the decomposition by a simple example. Consider the decision table in Figure 1. It relates the input attributes x_1, x_2, and x_3 to the class y, such that $y = F(x_1, x_2, x_3)$. There are three possible non-trivial partitions of attributes that yield three different decompositions $y = G_1(x_1, H_1(x_2, x_3))$, $y = G_2(x_2, H_2(x_1, x_3))$, $y = G_3(x_3, H_3(x_1, x_2))$. The first two are given in Figure 2, and the comparison shows that:

- decision tables in the decomposition $y = G_1(x_1, H_1(x_2, x_3))$ are smaller than those for $y = G_2(x_2, H_2(x_1, x_3))$,
- the new concept $c_1 = H_1(x_2, x_3)$ uses only three values, whereas that for $H_2(x_1, x_3)$ uses four,
- we found it hard to interpret decision tables G_2 and H_2, whereas by inspecting H_1 and G_1 it can be easy to see that $c_1 = MIN(x_2, x_3)$ and $y = MAX(x_1, c_1)$. This can be even more evident with the reassignment of c_1's values: 1 to `lo`, 2 to `med`, and 3 to `hi`.

The above comparison indicates that the decomposition $y = G_2(x_2, H_2(x_1, x_3))$ yields more complex and less interpretable decision tables than the decomposition $y = G_1(x_1, H_1(x_2, x_3))$.

x_1	x_2	x_3	y
lo	lo	lo	lo
lo	lo	hi	lo
lo	med	lo	lo
lo	med	hi	med
lo	hi	lo	lo
lo	hi	hi	hi
med	lo	lo	med
med	lo	hi	med
med	med	lo	med
med	med	hi	med
med	hi	lo	med
med	hi	hi	hi
hi	lo	lo	hi
hi	lo	hi	hi
hi	med	lo	hi
hi	med	hi	hi
hi	hi	lo	hi
hi	hi	hi	hi

Figure 1: An example decision table $y = F(x_1, x_2, x_3)$.

The questions of interest are thus:

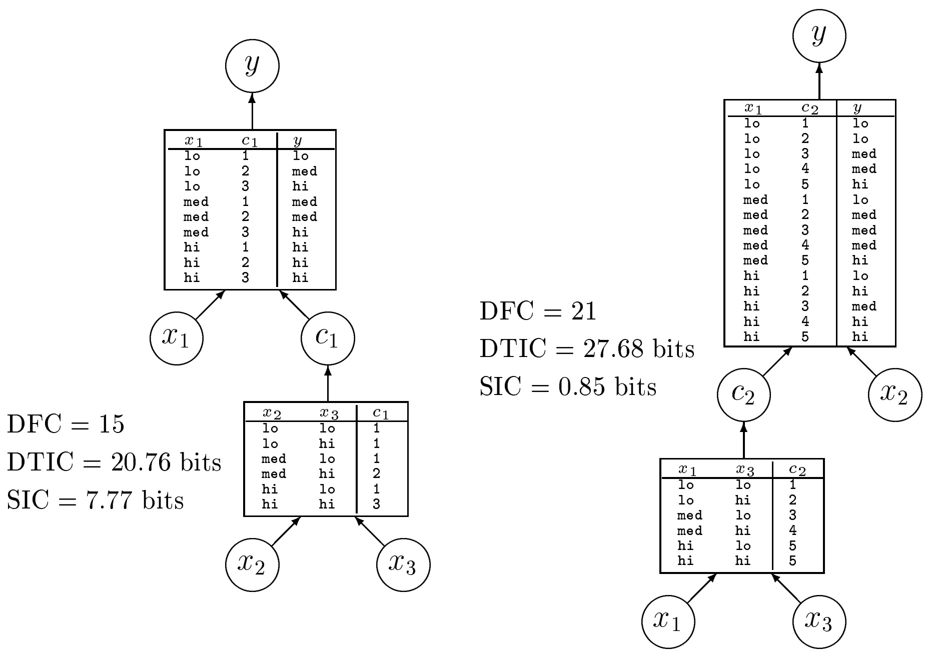
1. How do we measure the overall complexity of original decision table and of the decomposed system?

2. Which are the criteria that can guide the single decomposition step to chose among possible decompositions?

3. How much information is contained within the hierarchical structure itself?

4. How does interpretability relate to the overall complexity of decision tables in the decomposed system? Is a less complex system also easier to interpret?

Some of these questions were already addressed in the area of computer aided circuit design where decomposition is used to find a circuit of minimal complexity that implements a specific tabulated Boolean function. There, the methods mostly rely on the complexity and partition selection criterion known as Decomposed Function Cardinality (DFC, see [18]), but its appropriateness has already been questioned [14]. Furthermore, a question is whether this criterion can be used for the decomposition of decision tables of interest to machine learning, where attributes and classes usually take more than two values. Moreover, the main concern of Boolean function decomposition is the minimization of digital



x_1	c_1	y
lo	1	lo
lo	2	med
lo	3	hi
med	1	med
med	2	med
med	3	hi
hi	1	hi
hi	2	hi
hi	3	hi

x_2	x_3	c_1
lo	lo	1
lo	hi	1
med	lo	1
med	hi	2
hi	lo	1
hi	hi	3

x_1	c_2	y
lo	1	lo
lo	2	lo
lo	3	med
lo	4	med
lo	5	hi
med	1	lo
med	2	med
med	3	med
med	4	med
med	5	hi
hi	1	lo
hi	2	hi
hi	3	med
hi	4	hi
hi	5	hi

x_1	x_3	c_2
lo	lo	1
lo	hi	2
med	lo	3
med	hi	4
hi	lo	5
hi	hi	5

Figure 2: Two different decompositions of the decision table from Figure 1. Also given are the overall complexities of the decision tables and the information content of the structure (see Section 4 for definitions). Original decision table had DFC = 18 and DTIC = 28.53 bits.

circuit, leaving aside the question of comprehensibility and interpretability of the resulting hierarchy.

The paper is organized as follows. The next section reviews the related work on decision table decomposition with the emphasis on its use for machine learning. The decomposition algorithm to be used throughout the paper is presented in section 3. Section 4 introduces two new partition selection criteria that are based on information content of decision tables (DTIC) and on cardinality of newly discovered concepts (CM). That section also discusses how DFC and DTIC may be used to estimate the overall complexity of derived decision tables, and shows how DTIC may be used to assess information content of the discovered hierarchical structure itself. Section 5 experimentally evaluates the different criteria and complexity measures. Section 6 summarizes the results and concludes the paper.

2 Related work

The decomposition approach to machine learning was used early by a pioneer of artificial intelligence, A. Samuel. He proposed a method based on a *signature table system* [19] and successfully used it as an evaluation mechanism for checkers playing programs. This approach was later improved by Biermann et al. [3]. Their method, however, did not address the problem of deriving the structure of concepts and was mainly limited to cases where the training examples completely covered the attribute space.

A similar approach had been defined even earlier within the area of switching circuit design. In 1956, Ashenhurst reported on a unified theory of *decomposition of switching functions* [2]. The decomposition method proposed by Ashenhurst was essentially the same as that of Samuel and Biermann, except that it was used to decompose a truth table of a specific Boolean function to be then realized with standard binary gates. Most of other related work of those times is reported and reprinted in [7].

Recently, the Ashenhurst-Curtis approach was substantially improved by research groups of M. A. Perkowski, T. Luba, and T. D. Ross. In [15], Perkowski et al. report on the *decomposition approach for incompletely specified switching functions*. Luba [9] proposes a method for the decomposition of *multi-valued switching functions* in which each multi-valued variable is encoded by a set of Boolean variables. The authors identify the potential usefulness of function decomposition for machine learning, and Goldman [8] indicates that the decomposition approach to switching function design might be termed *knowledge discovery*, since a function not previously foreseen by the users might be discovered. From the viewpoint of machine learning, however, the main drawbacks of existing methods are that they are mostly limited to Boolean functions and incapable of dealing with noise.

Feature discovery has been at large investigated by *constructive induction* [11]. Perhaps closest to the function decomposition method are the constructive induction systems that use a set of existing attributes and a set of constructive operators to derive new attributes.

Several such systems are presented in [10, 16, 17].

Within machine learning, there are other approaches that are based on problem decomposition, but where the problem is decomposed by the expert and not induced by a machine. A well-known example is *structured induction*, developed by A. Shapiro [20]. His approach is based on a manual decomposition of the problem. For every intermediate concept either a special set of learning examples is used or an expert is consulted to build a corresponding decision tree. In comparison with standard decision tree induction techniques, Shapiro's approach exhibits about the same classification accuracy with the increased transparency and lower complexity of the developed models. Michie [12] emphasizes the important role the structured induction will have in the future development of machine learning and lists several real problems that were solved in this way.

The work presented here is based on our own decomposition algorithm [23] in which we took the approach of Curtis [7] and Perkowski [15], and extended it to handle multi-valued categorical attributes and functions. Although the algorithm lacks the noise-handling ability and is thus not fully equipped yet for a general machine learning task, it was demonstrated to perform well [23] in fairly complex problem domains with up to 15 nominal attributes.

3 Decision table decomposition algorithm

Let F be a decision table consisting of attribute-value vectors that map the attributes $X = \{x_1, \dots, x_n\}$ to the class y, so that $y = F(X)$. A single decomposition step searches through all the partitions of attributes X into a *free* set A and *bound* set B, such that $A \cap B = \emptyset$, $A \cup B = X$, and A and B contain at least one attribute. Let us denote such a partition with $A|B$ and assume that a *partition selection criterion* $\psi(A|B)$ exists that measures the appropriateness of this partition for decomposition (partitions with lower ψ are more appropriate). The partition with the lowest ψ is selected and F is decomposed to G and H, so that $y = G(A, c)$ and $c = H(B)$. Provided there exists a *complexity measure* θ for F, G, and H, F is decomposed only if the *complexity condition* $\theta(F) > \theta(G) + \theta(H)$ is satisfied. Several partition selection (ψ) and complexity (θ) measures are introduced in the next section.

An algorithm that implements the single decomposition step and decomposes a decision table F to G and H is in detail described in [23]. Here, we illustrate it informally using the decision table in Figure 1. For every partition of attributes, the algorithm constructs a partition matrix with attributes of bound set in columns and of free set in rows. Each column in the partition matrix denotes the behavior of F for a specific combination of values of bound attributes. Same columns can then be represented with the same value of c, and the number of different columns is equal to the minimal number of values for c to be used for decomposition. In this way, every column is then assigned a value of c, and G and

H are straightforwardly derived from such annotated partition matrix. For each of three partitions for our sample decision table F, the partition matrices with the corresponding values of c are given in Figure 3.

x_1 \ x_2 / x_3	lo / lo	lo / hi	med / lo	med / hi	hi / lo	hi / hi
lo	lo	lo	lo	med	lo	hi
med	med	med	med	med	med	hi
hi	hi	hi	hi	hi	hi	hi
c	1	1	1	2	1	3

x_2 \ x_1 / x_3	lo / lo	lo / hi	med / lo	med / hi	hi / lo	hi / hi
lo	lo	lo	med	med	hi	hi
med	lo	med	med	med	hi	hi
hi	lo	hi	med	hi	hi	hi
c	1	2	3	4	5	5

x_3 \ x_1 / x_2	lo / lo	lo / med	lo / hi	med / lo	med / med	med / hi	hi / lo	hi / med	hi / hi
lo	lo	lo	lo	med	med	med	hi	hi	hi
hi	lo	med	hi	med	med	hi	hi	hi	hi
c	1	2	3	4	5	5	6	6	6

Figure 3: Partition matrices for three different partitions of attributes x_1, x_2, and x_3 of decision table in Figure 1.

The assignment of values of c for each column is trivial if decision table instances completely cover the attribute space. If this is not the case, Wan and Perkowski [21] proposed an approach that treats missing decision table entries as "don't cares". Each partition matrix can then have several assignments of values for c. The problem of finding the assignment that uses the fewest values is then equivalent to optimal graph coloring. Graph coloring is an NP-hard problem and the computation time of an exhaustive search algorithm is prohibitive even for small graphs. Instead, Wan and Perkowski [21] suggested a heuristic Color Influence Method of polynomial complexity and showed that the method performed well compared to the optimal algorithm. Although the examples used in this paper use decision tables that completely cover the attribute space, the complexity and partition measures introduced apply with no difference to incompletely covered cases as well.

The decomposition algorithm examines all the decision tables in the evolving structure and then applies a single decomposition step to the decision table and its partition that was evaluated as the most appropriate by ψ and that satisfies the complexity condition $\theta(F) > \theta(G) + \theta(H)$. If several partitions are scored equal, the algorithm arbitrarily selects one among those with the lowest number of elements in the bound set. The process is repeated until no decomposition is found that would satisfy the complexity condition.

We illustrate this stepwise decomposition using the CAR domain that is described in section 5. Figure 4 shows a possible evolving hierarchical structure obtained by decomposition. Each consecutive structure is a result of a single decomposition step. Only the structure without decision tables is shown. Compared to the original hierarchical model, c2 corresponds to price, c4 to technical characteristics, and c1 to comfort. c3 was not used in the original model.

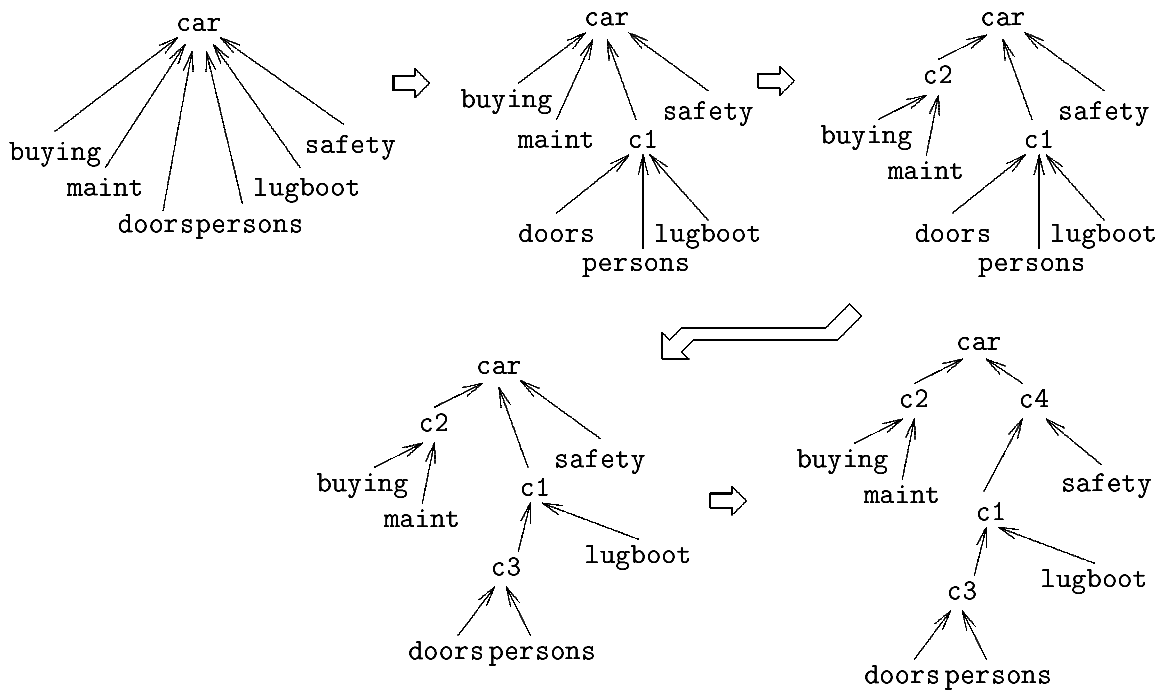


Figure 4: Evolving hierarchy discovered by decomposition of the decision table in the CAR domain. Each consecutive structure results from a single-step decomposition of its predecessor.

The overall time complexity of decision table decomposition algorithm is polynomial in the number of examples, number of attributes, and maximal number of columns in partition matrices [24]. As the latter grows exponentially with the number of bound attributes, it is advantageous to limit the size of the bound set. In the experiments presented in Section 5, however, the problems were sufficiently small to examine all possible bound sets.

The above decomposition algorithm was implemented in the C language as a part of the system called HINT (Hierarchy INduction Tool) that runs on several UNIX platforms, including HP/UX and SGI Iris [22].

4 Partition selection criteria and complexity measures

This section reviews one and introduces two new partition selection criteria $\psi(A|B)$. For each, it also defines the complexity measure and corresponding complexity condition. Furthermore, two overall complexity measures for the hierarchy of decision tables are defined, and finally, a measure for estimating the information content of the hierarchy itself is presented.

4.1 Partition selection criteria

4.1.1 Decomposed function cardinality (DFC)

This measure was originally proposed by Abu-Mostafa [1] as a general measure of complexity and used in decomposition of Boolean functions (see [18]). DFC is based on the cardinality of the function. Given a decision table $F(X)$, DFC-based complexity is defined as:

$$\theta_{\mathrm{DFC}}(F) = ||X|| = \prod_i |x_i|,\ x_i \in X \tag{1}$$

where $|x_i|$ represents the cardinality of attribute x_i, i.e., the number of values it uses.

The DFC partition selection criterion for decomposition $F(X) = G(A, c)$ and $c = H(B)$ is then:

$$\psi_{\mathrm{DFC}}(A|B) = \theta_{\mathrm{DFC}}(G) + \theta_{\mathrm{DFC}}(H) = |c|\ ||A|| + ||B|| \tag{2}$$

The complexity condition using the above definitions is $\theta_{\mathrm{DFC}}(F) > \theta_{\mathrm{DFC}}(G)+\theta_{\mathrm{DFC}}(H)$, or equivalently $||X|| > |c|\ ||A|| + ||B||$.

For the decision table in Figure 1 and partition matrices in Figure 3, the partition selection criteria are: $\psi_{\mathrm{DFC}}(x_1|x_2x_3) = 9 + 6 = 15$, $\psi_{\mathrm{DFC}}(x_2|x_1x_3) = 15 + 6 = 21$, and $\psi_{\mathrm{DFC}}(x_3|x_1x_2) = 12 + 9 = 21$. $\theta_{\mathrm{DFC}}(F)$ is 18. The only partition that satisfies the DFC decomposition criterion is $x_1|x_2x_3$.

Although its ability to guide the decomposition of Boolean functions has been illustrated in several references including [18], DFC has been recently criticized by Perkowski and Grygiel for deficiencies in handling some classes of functions including multi-output symmetric functions [14]. Moreover, we are not aware of any study of the applicability of DFC measures for decomposition of decision tables other than Boolean.

4.1.2 Information content of decision tables (DTIC)

This measure is based on the idea of Biermann et al. [3] who counted the number of different functions that can be represented by a given *signature table schema*, i.e., a tree structure of concepts whose cardinality is predefined.

A decision table $y = F(X)$ can represent $|y|^{||X||}$ different functions and this number corresponds to the cardinality of the function. The information such decision table contains is then equal to

$$\theta_{\mathrm{DTIC}}(F) = ||X|| \log_2 |y| \ \ \text{bits} \tag{3}$$

Note that for binary functions where $|y| = 2$, this is equal to $\theta_{\mathrm{DFC}}(F)$.

When decomposing $y = F(X)$ to $y = G(A, c)$ and $c = H(B)$, we assign a single value from the set $\{1, 2, \ldots, |c|\}$ to each of the columns of partition matrix $A|B$. But, each of the values have to be assigned to at least one instance. In other words, from $|y|^{||B||}$ different functions we have to subtract all those that use less than $|c|$ values. The number of different functions with exactly $|c|$ possible values is therefore $N(|c|)$, where N is defined as:

$$\begin{aligned} N(x) = &\ x^{||B||} - \sum_{i=1}^{x-1} \binom{x}{i} N(i) \\ N(1) = &\ 1 \end{aligned} \tag{4}$$

Furthermore, since the actual label (value of c) of the column is unimportant, there are $|c|!$ such equivalent assignments and therefore $|c|!$ equivalent decision tables H. H therefore uniquely represents $N(|c|)/|c|!$ functions with exactly $|c|$ values, and the corresponding information content is:

$$\theta'_{\mathrm{DTIC}}(H) = \log_2 N(|c|) - \log_2(|c|!) \ \text{ bits} \tag{5}$$

The DTIC partition selection criterion prefers the decompositions with simple decision tables G and H and therefore low information content, so that:

$$\psi_{\mathrm{DTIC}}(A|B) = \theta_{\mathrm{DTIC}}(G) + \theta'_{\mathrm{DTIC}}(H) \tag{6}$$

The DTIC-based complexity condition is:

$$\theta_{\mathrm{DTIC}}(F) > \theta_{\mathrm{DTIC}}(G) + \theta'_{\mathrm{DTIC}}(H) \tag{7}$$

For the decision table in Figure 1, DTIC criteria evaluate to: $\psi_{\mathrm{DTIC}}(x_1|x_2x_3) = 20.76$ bits, $\psi_{\mathrm{DTIC}}(x_2|x_1x_3) = 27.68$ bits, and $\psi_{\mathrm{DTIC}}(x_3|x_1x_2) = 30.39$ bits. $\theta_{\mathrm{DTIC}}(F)$ is 28.53 bits, and, in contrast to DFC, two partitions qualify for decomposition. Among these, as with DFC, $x_1|x_2x_3$ has the lowest θ_{DTIC}.

4.1.3 Column multiplicity (CM)

This is the simplest complexity measure introduced in this paper and equals to the cardinality of c ($|c|$), also referred to by Ashenhurst and Curtis as *column multiplicity* number of partition matrix [2, 7]. The idea for this measure came from practical experience with DEX decision support system [5]. There, the hierarchical system of decision tables is constructed manually and it has been found that decision tables with small number of output values are easier to construct and interpret. Formally,

$$\psi_{\mathrm{CM}}(A|B) = |c| \tag{8}$$

As a CM complexity condition a bound on $|c|$ might be used, but such bound would have to be domain dependent. Instead, we use DTIC complexity condition when using CM.

For our example and similarly to DFC and DTIC, CM also selects the partition $x_1|x_2x_3$ with $\psi_{\mathrm{CM}} = 3$. Other two partitions have $\psi_{\mathrm{CM}}(x_2|x_1x_3) = 5$ and $\psi_{\mathrm{CM}}(x_3|x_1x_2) = 6$.

4.2 Complexity estimation for decision table hierarchy

Using DFC, the overall complexity of decision tables in the hierarchical discovered structure is the sum of θ_{DFC} for each decision table.

For DTIC, the complexity estimation again uses the sum of DTIC complexities of each of the decision tables, with the distinction that θ_{DTIC} is used for the decision table at the root of the hierarchy and θ'_{DTIC} for all other decision tables.

Note that the so-obtained DTIC complexity estimation is just an approximation of the exact complexity that would take into account the actual number of functions representable by a multi-level hierarchy. This is because DTIC is designed for a single table only and does not take into account the reducibility [3] that occurs in multi-level hierarchies and effectively decreases the number of representable functions. Therefore, the estimated overall DTIC is the upper bound of the actual complexity.

4.3 Structure information content (SIC)

Using DTIC we can assess both the amount of information contained in the original decision table and contained in the resulting decision tables that were constructed by decomposition. The difference of the two is the information contained in the hierarchy itself. We call this measure *structure information content* (SIC). The more informative is the hierarchy, the overall less complex are the resulting decision tables.

5 Experimental evaluation

To evaluate the proposed partition selection and complexity measures, we used three artificial and three real-world domains that were selected so that the hierarchical structure was either known in advance or could have been easily anticipated. For each domain, the decomposition aimed to discover this structure. For evaluation, we qualitatively assess the similarity of the two structures and quantitatively compare them by using the proposed complexity measures.

Each of six domains is represented with the initial decision table with instances that completely cover the attribute space. The experiments could as well be done with sparser decision tables (see [23]), but in these experiments we wanted to concentrate on complexity issues rather than generalization. Namely, the proposed partition measures depend only on cardinalities of attributes and concepts, and not on the actual number of instances in decision tables. XXXXXXX The results of decompositions are shown as hierarchy graphs, where, unless otherwise noted, the labels of intermediate concepts indicate the order in which they were discovered by DTIC and DFC-guided decompositions.

5.1 Boolean function $y = (x_1 \text{ OR } x_2) \text{ AND } x_3 \text{ AND } (x_4 \text{ XOR } x_5)$

For this case, the initial decision table has $2^5 = 32$ instances, $\theta_{\text{DFC}} = 32$ and $\theta_{\text{DTIC}} = 32$ bits. While decomposition with DTIC and CM discovered the anticipated structure, the DFC-guided decomposition terminated too soon because the complexity condition did not allow to decompose the decision tables any further (see Figure 5). Note that overall DFC is the same for DTIC, CM, and DFC discovered decision tables, while the structure information content is higher for those of DTIC and CM. The decision tables (not shown in the figure) where checked for interpretability and were found to represent the expected functions.

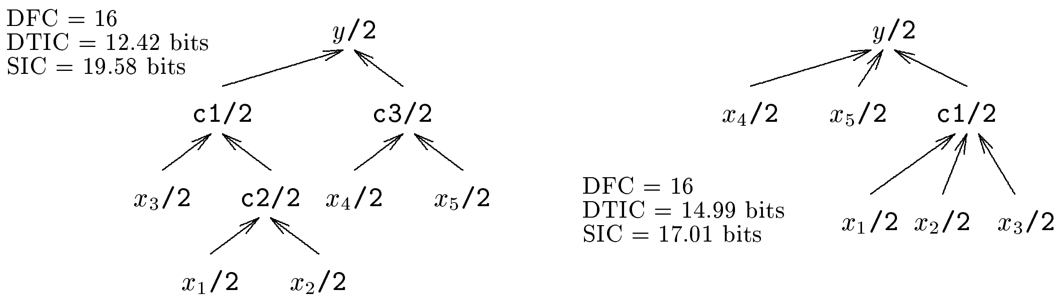


Figure 5: Decomposition of decision table representing the function $y = (x_1 \text{ OR } x_2)$ AND x_3 AND $(x_4 \text{ XOR } x_5)$ guided by DTIC and CM (left), and DFC (right).

5.2 Palindrome $y = \text{PAL}(x_1, x_2, x_3, x_4, x_5, x_6)$

This function returns 1 if the string $x_1 \ldots x_6$ is a palindrome and returns 0 otherwise, i.e., $y = (x_1 = x_6)$ AND $(x_2 = x_5)$ AND $(x_3 = x_4)$. In the first experiment, the Boolean attributes $x_1 \ldots x_6$ were used. The initial decision table has $\theta_{\text{DFC}} = 64$ and $\theta_{\text{DTIC}} = 64$ bits. Again, the decomposition with DFC stops sooner and the domain favors the decomposition using CM and DTIC. However, for both this and previous case a DFC-guided decomposition could discover the expected structure if the corresponding complexity condition would be changed to $\theta_{\text{DFC}}(F) \geq \theta_{\text{DFC}}(G) + \theta_{\text{DFC}}(H)$. Using this condition, the decomposition can proceed even if the joint complexity of G and H is equal to the one of F.

The same experiment was repeated with three-valued attributes $x_1 \ldots x_6$. This time, however, all three criteria lead to the same and anticipated hierarchical structure, which is again the same as the left one in Figure 6.

5.3 Function $y = \text{MIN}(x_1, \text{AVG}(x_2, \text{MAX}(x_3, x_4), x_5))$

The ordinal attributes $x_1 \ldots x_5$ can take values 1,2, and 3. While MIN and MAX have standard interpretation, AVG computes the average of its arguments and rounds it to the

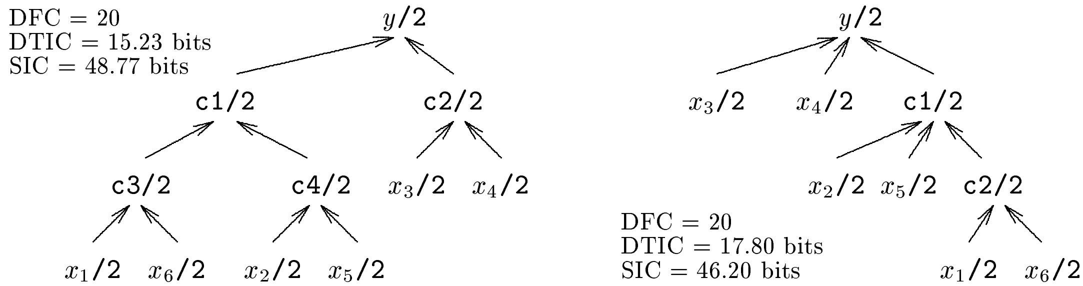


Figure 6: Decomposition of decision table representing the palindrome function guided by DTIC and CM (left), and DFC (right).

closest integer. The initial decision table has $\theta_{\text{DFC}} = 243$ and $\theta_{\text{DTIC}} = 385.15$ bits. The anticipated and discovered structures are shown in Figure 7. Quite surprisingly, in all three cases the decomposition yields a structure with higher structure information content than that of expected structure by introducing an additional five-valued intermediate concept. If this is removed, the discovered structure and decision tables would be the same as anticipated.

It is also interesting to note that the structure discovered using CM on one side and DFC or DTIC on the other are different but of the same complexity. This example illustrates that for a specific domain there may exist several optimal structures with regard to complexity.

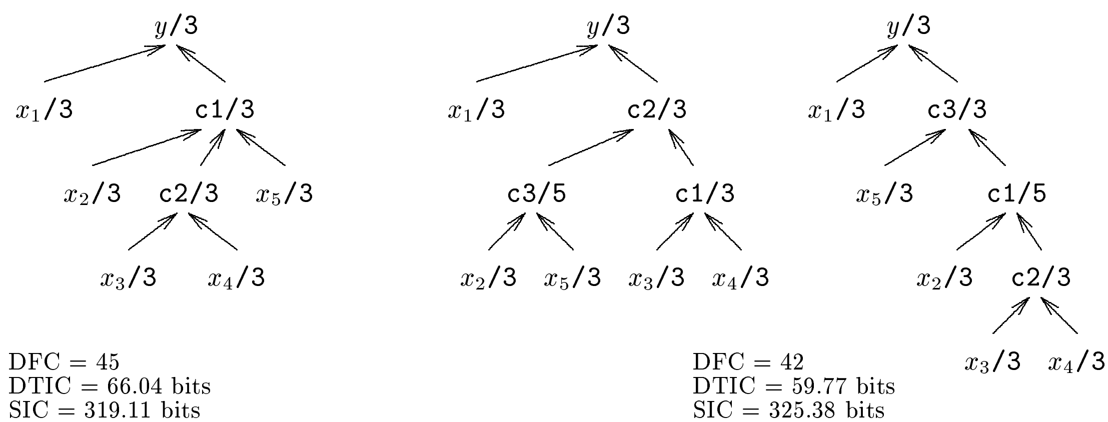


Figure 7: Decompositions of the function $y = \text{MIN}(x_1, \text{AVG}(x_2, \text{MAX}(x_3, x_4), x_5))$: the anticipated structure (left), the structure discovered using CM (middle), and DFC and DTIC (right). The complexity and information measures for the latter two decompositions are the same.

5.4 DEX models CAR, EMPLOY, and NURSERY

An area where concept hierarchies have been used extensively is decision support. There, the problem is to select an option from a set of given options so that it best satisfies the aims or goals of the decision maker. DEX [5] is a multi-attribute decision support system that has been extensively used to solve realistic decision making problems. DEX uses categorical attributes and expects the structure and the functions to be given by the expert. The formalism used to describe the resulting model and its interpretation is essentially the same as those derived through the decomposition process described in this paper. This makes models developed by DEX an ideal benchmark for the evaluation of decision table decomposition. In this paper, we use the following three DEX models:

CAR: A model for evaluating cars based on their price and technical characteristics. This simple model was developed for educational purposes and is described in [4].

EMPLOY: This is a simplified version of the models that were developed with DEX for a common problem of personnel management: selecting the best candidate for a particular job. While the realistic models that were practically used in several mid- to large-size companies in Ljubljana and Sarajevo consisted of more than 40 attributes, the simplified version uses only 7 attributes and 3 intermediate concepts and was presented in [6].

NURSERY: This model was developed in 1985 [13] to rank applications for nursery schools. It was used during several years when there was excessive enrollment to these schools in Ljubljana, and the rejected applications frequently needed an objective explanation. The final decision depended on three subproblems: (1) occupation of parents and child's nursery, (2) family structure and financial standing, and (3) social and health picture of the family.

The goal of this experiment was to reconstruct these models from examples. The learning examples were derived from the original models, where for all combinations of input attributes the class was determined by a corresponding original model.

The discovered hierarchies are given in Figures 8, 9, and 10. In all cases, DFC, DTIC, and CM-guided decomposition found the same hierarchical structures and corresponding decision tables. Using DFC and DTIC, the order in which new intermediate concepts were found was the same but different to the one when CM was used. For example, in EMPLOY, DFC and DTIC-guided decomposition discovered `c1` first, while, using CM, this concept was discovered as the last one.

All the discovered hierarchies have higher information content than the original ones. Also, the overall complexity of decision tables is lower according to both DFC and DTIC. Most importantly, the discovered structures are very similar to the original ones. In fact, if `c3` would be removed from CAR (making `c4` directly dependent on `lugboot`, `doors`, and persons), the two structures would be the same. The same applies to EMPLOY and NURSERY if c1 and c2 are removed, respectively. In other words, the decomposition found the same structures as the original ones but additionally decomposed the decision tables for comfort (CAR), employ (EMPLOY), and struct+finan (NURSERY) and in this way obtained less complex decision tables.

The derived decision tables were compared to the original ones and found to be the same but in the names used for instance labels (decomposition uses numbers while original decision tables use meaningful names). The exception are decision tables for tech and comfort in the CAR domain, where the decomposition found redundant values of comfort. If these values were removed from the original model, the corresponding decision tables would have been again the same.

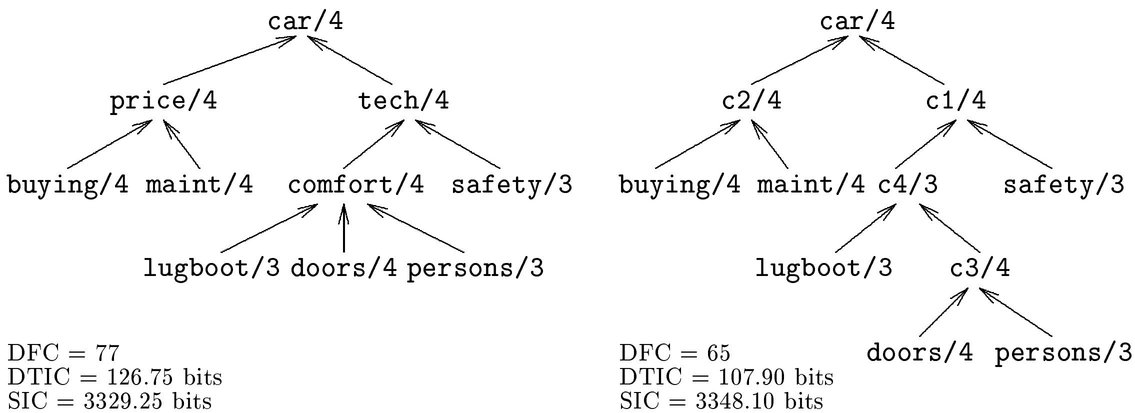


Figure 8: The original concept hierarchy of CAR (left) and the decompositions based on CM, DFC and DTIC (right).

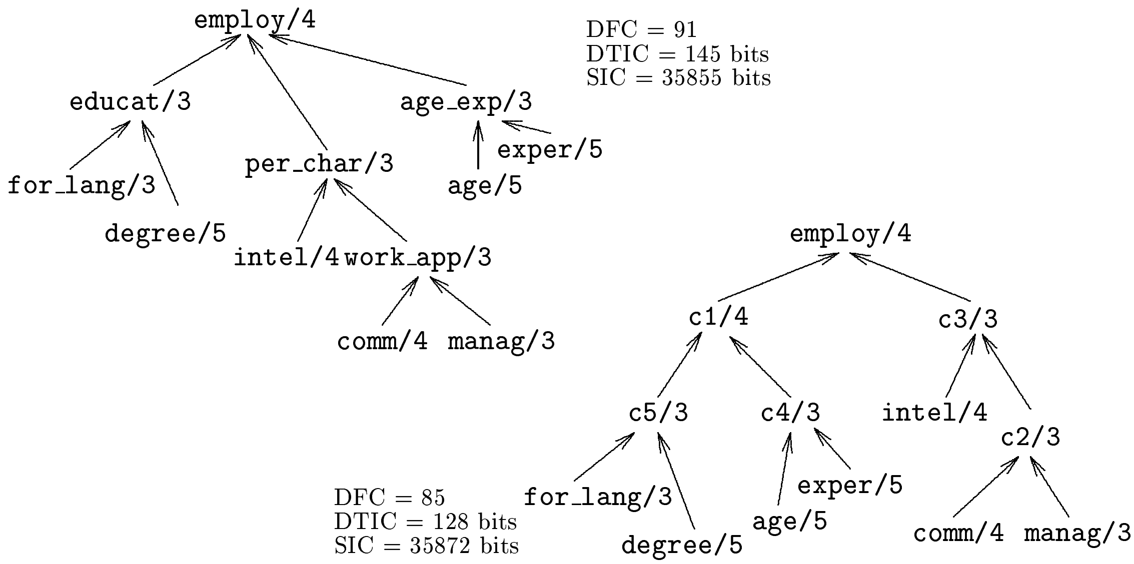


Figure 9: The original concept hierarchy of EMPLOY (top) compared to the hierarchy discovered by CM, DFC, and DTIC-guided decomposition (bottom).

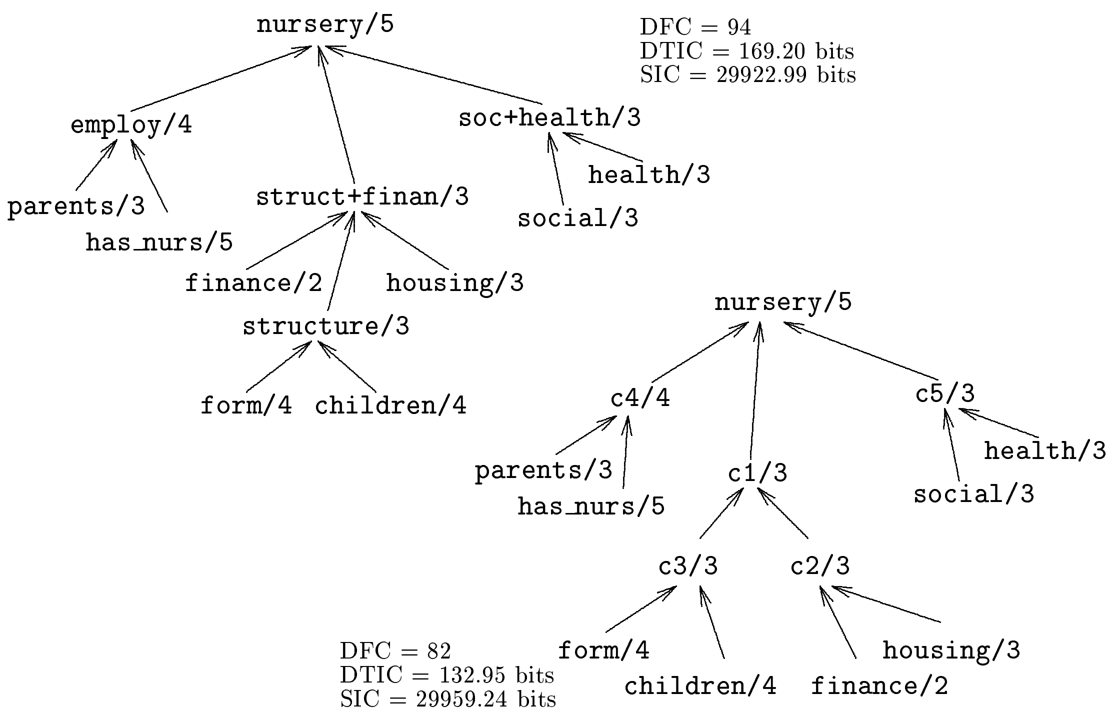


Figure 10: The original (left) and discovered concept hierarchy using CM, DFC and DTIC criteria (right) for NURSERY.

6 Conclusion

The paper investigates the appropriateness of three partition selection for decision table decomposition: decision table information content (DTIC) and column multiplicity (CM) introduced in this paper and decomposed function cardinality (DFC) that has already been used primarily for the decomposition of Boolean functions.

The experimental evaluation exposes the deficiency of DFC when decomposing a decision table that expresses a Boolean function. This may be alleviated by relaxing the DFC complexity condition. Furthermore, in more complex domains with multi-valued attributes, the decomposition guided by any of the proposed criteria discovered equal or better hierarchical structures than expected with respect to the complexity of decision tables and informativity of the hierarchical structure. The order under which the intermediate concepts were discovered was the same for DFC and DTIC, but different for CM. A qualitative evaluation of derived structures reveals that in general the discovered decision tables represent meaningful and interpretable concepts.

Although less complex in definition and easier to compute, DFC and CM both stand well in comparison with more complex partition selection measure DTIC. Also comparable is the utility of DFC and DTIC to assess the complexity of the original and derived decision tables, although we have shown that DFC-based measure performed worse on two Boolean functions. Overall, while DFC and DTIC have better theoretical foundations than an intuitive partition selection measure CM, the experimental evaluation does not indicate that any of these is to be strictly preferred over the other.

The decision table decomposition was primarily developed for switching circiut deasign. However, experiments in non-trivial domains like DEX's strongly encourage further research and development of this method for machine learning and knowledge discovery. To become a general machine learning tool, the method will have to be extended so as to handle continuous attributes and deal with noise in learning examples.

References

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

[2] R. L. Ashenhurst. The decomposition of switching functions. Technical report, Bell Laboratories BL-1(11), pages 541–602, 1952.

[3] A. W. Biermann, J. Fairfield, and T. Beres. Signature table systems and learning. *IEEE Trans. Syst. Man Cybern.*, 12(5):635–648, 1982.

[4] M. Bohanec and V. Rajkovič. Knowledge acquisition and explanation for multi-attribute decision making. In *8th Intl Workshop on Expert Systems and their Applications*, pages 59–78, Avignon, France, 1988.

[5] M. Bohanec and V. Rajkovič. DEX: An expert system shell for decision support. *Sistemica*, 1(1):145–157, 1990.

[6] M. Bohanec, B. Urh, and V. Rajkovič. Evaluating options by combined qualitative and quantitative methods. *Acta Psychologica*, 80:67–89, 1992.

[7] H. A. Curtis. *A New Approach to the Design of Switching Functions.* Van Nostrand, Princeton, N.J., 1962.

[8] J. Goldman. Pattern theoretic knowledge discovery. Technical report, GSRP Wright Laboratories, 1994.

[9] T. Luba. Decomposition of multiple-valued functions. In *25th Intl. Symposium on Multiple-Valued Logic*, pages 256–261, Bloomigton, Indiana, May 1995.

[10] R. S. Michalski. A theory and methodology of inductive learning. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134. Kaufmann, Paolo Alto, CA, 1983.

[11] R. S. Michalski. Understanding the nature of learning: issues and research directions. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 3–25. Kaufmann, Los Atlos, CA, 1986.

[12] D. Michie. Problem decomposition and the learning of skills. In N. Lavrač and S. Wrobel, editors, *Machine Learning: ECML-95*, Notes in Artificial Intelligence 912, pages 17–31. Springer-Verlag, 1995.

[13] M. Olave, V. Rajkovič, and M. Bohanec. An application for admission in public school systems. In I. Th. M. Snellen, W. B. H. J. van de Donk, and J.-P. Baquiast, editors, *Expert Systems in Public Administration*, pages 145–160. Elsevier Science Publishers (North Holland), 1989.

[14] M. A. Perkowski. A survey of literature on function decomposition. Technical report, GSRP Wright Laboratories, Ohio OH, October 1995.

[15] M. A. Perkowski et al. Unified approach to functional decompositions of switching functions. Technical report, Warsaw University of Technology and Eindhoven University of Technology, 1995.

[16] B. Pfahringer. Controlling constructive induction in CiPF. In F. Bergadano and L. De Raedt, editors, *Machine Learning: ECML-94*, pages 242–256. Springer-Verlag, 1994.

[17] H. Ragavan and L. Rendell. Lookahead feature construction for learning hard concepts. In *Proc. Tenth International Machine Learning Conference*, pages 252–259. Morgan Kaufman, 1993.

[18] T. D. Ross, M. J. Noviskey, D. A. Gadd, and J. A. Goldman. Pattern theoretic feature extraction and constructive induction. In *Proc. ML-COLT '94 Workshop on Constructive Induction and Change of Representation*, New Brunswick, New Jersey, July 1994.

[19] A. Samuel. Some studies in machine learning using the game of checkers II: Recent progress. *IBM J. Res. Develop.*, 11:601–617, 1967.

[20] A. D. Shapiro. *Structured induction in expert systems*. Turing Institute Press in association with Addison-Wesley Publishing Company, 1987.

[21] W. Wan and M. A. Perkowski. A new approach to the decomposition of incompletely specified functions based on graph-coloring and local transformations and its application to FPGA mapping. In *Proc. of the IEEE EURO-DAC '92*, pages 230–235, Hamburg, September 1992.

[22] B. Zupan. *Machine learning based on function decomposition*. PhD thesis, University of Ljubljana, April 1997. Available at `http://www-ai.ijs.si/BlazZupan/papers.html`.

[23] B. Zupan and M. Bohanec. Learning concept hierarchies from examples by function decomposition. Technical report, IJSDP–7455, J. Stefan Institute, Ljubljana, September 1996. URL `ftp://ftp-e8.ijs.si/pub/reports/IJSDP-7455.ps`.

[24] B. Zupan, M. Bohanec, I. Bratko, and J. Demšar. Machine learning by function decomposition. In Jr. D. H. Fisher, editor, *Proc. Fourteenth International Conference on Machine Learning (ICML-97)*, pages 421–429, San Mateo, CA, 1997. Morgan Kaufmann.