Experimental evaluation of three partition selection criteria for decision table decomposition

Blaž Zupan and Marko Bohanec
Jožef Stefan Institute,
Jamova 39, Ljubljana, Slovenia
Phone: +386 61 177 3900
Fax: +386 61 125 1038
E-mail: blaz.zupan@ijs.si, marko.bohanec@ijs.si

Keywords: decision table decomposition, partition selection criteria, intermediate concepts, concept hierarchy, knowledge discovery

Edited by: Rudi Murn
Received: May 20, 1997 **Revised:** March 7, 1998 **Accepted:** April 18, 1998

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 article introduces two such criteria and experimentally compares their performance with a criterion originally used for the decomposition of Boolean functions. 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.

1 Introduction

A decision table provides a simple means for concept representation. It represents a concept with labeled instances, each relating a set of attribute values to a class. 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 variable. F, G and H are functions represented by *decision tables*, i.e., possibly incomplete sets of attribute-value vectors with assigned classes. A and B are nonempty subsets of input attributes such that $A \cup B = X$. The functions G and H are developed by decomposition 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 G and H, the result in general is a *hierarchy* of decision tables. As each decision table represents a concept, the result of decomposition can be regarded also as a *concept hierarchy*.

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 (Table 1). The decision table 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 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 1, and the comparison shows that:

- decision tables in the decomposition $y = G_1(x_1, H_1(x_2, x_3))$ are overall 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 five,
- it is hard to interpret decision tables G_2 and H_2,

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

Table 1: An example decision table.

whereas by inspecting G_1 and H_1 it can be easy to see that $c_1 = \text{MIN}(x_2, x_3)$ and $y = \text{MAX}(x_1, c_1)$. This can be even more evident with the reassignment of c_1's values: 1 to lo, 2 to med, and 3 to hi.

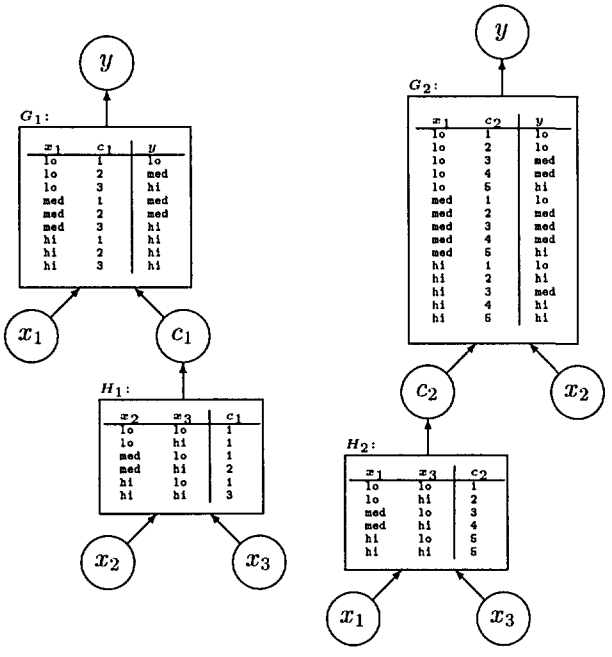

Figure 1: Two different decompositions of the decision table from Table 1.

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 $y = G_1(x_1, H_1(x_2, x_3))$. 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 [21]). However, 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 circuit, leaving aside the question of comprehensibility and interpretability of the resulting hierarchy.

This article is organized as follows. The next section reviews related work on decision table decomposition with the emphasis on its use for machine learning. The decomposition algorithm to be used throughout the article is presented in section 3. Section 4 introduces two new partition selection criteria that are based on the information content of decision tables (DTIC) and on the 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 the 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 article.

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* [22] 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 hierarchy of concepts, which was supposed to be given by a domain expert.

A similar approach had been defined even earlier within the area of switching circuit design. In 1956,

R.L. Ashenhurst reported on a unified theory of *decomposition of switching functions* [2]. The decomposition method proposed by Ashenhurst was used to decompose a completely specified truth table of a Boolean function to be then realized with standard binary gates. Thus, the method could construct concept hierarchies as well as their corresponding decision tables. Most of other related work of those times is reported and reprinted by Curtis [8].

Recently, the Ashenhurst-Curtis approach was substantially improved by research groups of M. A. Perkowski, T. Luba, and T. D. Ross. In [18], Perkowski et al. report on the *decomposition approach for incompletely specified switching functions.* Luba [12] proposed a method for the decomposition of *multi-valued switching functions* in which each multi-valued variable is encoded by a set of Boolean variables. A decomposition of k-valued functions was proposed by Files et al. [10]. The authors identify the potential usefulness of function decomposition for machine learning, and Goldman [11] indicates that the decomposition approach to switching function design might be termed *knowledge discovery,* since a function not previously foreseen might be discovered. From the viewpoint of machine learning, however, the main drawbacks of these 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* [14]. Perhaps closest to function decomposition 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 [13, 19, 20].

Within machine learning, there are other approaches that are based on problem decomposition, but where the problem is decomposed by the expert and not by a machine. A well-known example is *structured induction,* developed by Shapiro [23]. 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 [15] 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 [25] in which we took the approach of Curtis [8] and Perkowski et al. [18], and extended it to handle multi-valued categorical attributes and functions. The algorithm was demonstrated to perform well in terms of generalization [26], discovery of relevant concept hierarchies [7], and feature construction [27] in fairly complex problem domains.

3 Decomposition algorithm

Let F be a decision table consisting of attribute-value vectors that map the attributes $X = \{x_1, \ldots, x_n\}$ 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 each 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.

The algorithm that implements the single decomposition step and decomposes a decision table F to G and H is described in detail in [25]. Here, we illustrate it informally using the decision table from Table 1. For every attribute partition, the method constructs a *partition matrix* with the 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. The same columns can be represented with the same value of c. 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 assigned a value of c, and G and H are straightforwardly derived from such an annotated partition matrix. For each of three partitions for our example decision table F, the partition matrices with the corresponding values of c are given in Figure 2.

The assignment of c's values is trivial when decision table instances completely cover the attribute space. When this is not the case, Wan and Perkowski [24] 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 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 article use decision tables that completely cover the attribute space, the complexity and partition measures

x_1 \ x_2	lo	lo	med	med	hi	hi
x_3	lo	hi	lo	hi	lo	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	lo	lo	med	med	hi	hi
x_3	lo	hi	lo	hi	lo	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	lo	lo	lo	med	med	med	hi	hi	hi
x_2	lo	med	hi	lo	med	hi	lo	med	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 2: Partition matrices for Table 1 using three different partitions of attributes x_1, x_2, and x_3.

introduced apply with no difference to incompletely covered cases as well.

The decomposition algorithm examines all decision tables in the evolving concept hierarchy 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 3 shows a possible evolving concept hierarchy obtained by decomposition. Each consecutive hirarchy is a result of a single decomposition step. Only the hierarchical structure without decision tables is shown.

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 [26]. 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) [25]. HINT runs on several UNIX platforms, including HP/UX and SGI Iris.

4 Partition selection criteria and complexity measures

This section reviews one and introduces two new partition selection criteria. 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

Decomposed function cardinality (DFC) was originally proposed by Abu-Mostafa [1] as a general measure of complexity and used in decomposition of Boolean functions [21]. DFC is based on the cardinality of the function. Given a decision table $F(X)$, DFC-based complexity is defined as:

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

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:

$$\begin{aligned} \psi_{\mathrm{DFC}}(A|B) &= \theta_{\mathrm{DFC}}(G) + \theta_{\mathrm{DFC}}(H) \\ &= |c|\ ||A|| + ||B|| \end{aligned} \qquad (2)$$

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 our example decision table (Table 1) and the corresponding partition matrices (Figure 2), 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 $x_1|x_2x_3$.

DFC's ability to guide the decomposition of Boolean functions has been illustrated in several references including [21, 11]. For multi-valued logic synthesis, a DFC-guided decomposition was proposed in [10].

4.1.2 Information content of decision tables

Decision table information content (DTIC) is based on the idea of Biermann et al. [3] who counted the num-

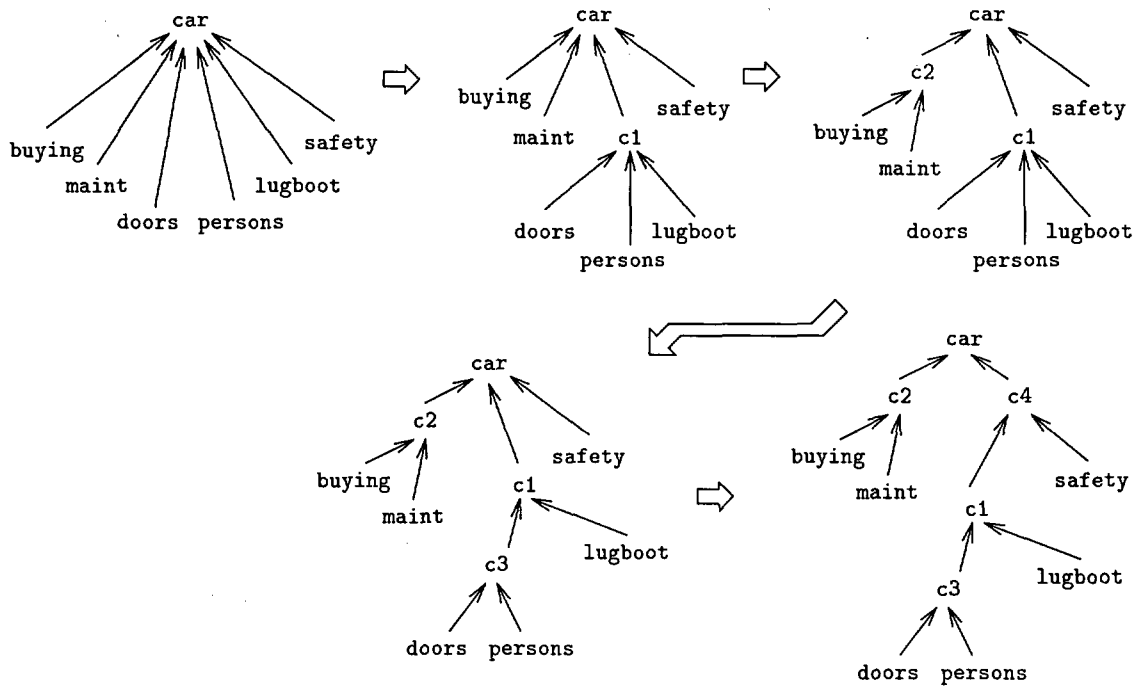

Figure 3: Evolving concept hierarchy discovered by decomposition of the CAR decision table. Each consecutive hierarchy results from a single-step decomposition of its predecessor.

ber of different functions that can be represented by a given *signature table schema*, i.e., a tree of concepts whose cardinality is predefined.

A decision table $y = F(X)$ can represent $|y|^{||X||}$ different functions. Assuming the uniform distribution of functions, the number of bits to encode such a decision table is then

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

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. But, each of the values has 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 not important, there are $|c|!$ such equivalent assignments and therefore $|c|!$ equivalent decision tables H. A specific H therefore uniquely represents $N(|c|)/|c|!$ functions with exactly $|c|$ values, and the corresponding information content is:

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

The DTIC partition selection criterion prefers the decompositions with simple decision tables G and H and 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 Table 1, DTIC evaluates 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, the partition $x_1|x_2x_3$ is preferred.

4.1.3 Column multiplicity

Column multiplicity (CM) is the simplest complexity measure introduced in this article and equals to the cardinality of c ($|c|$), also referred to by Ashenhurst and Curtis as *column multiplicity* number of partition matrix [2, 8]. Formally,

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

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.

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

Unlike DTIC and DFC, CM can not be simply summed up to determine the joint complexity of a set of decision tables, which is needed to determine the complexity condition. Consequently, when we employ CM to guide the partition selection, we use DTIC to determine the decomposability.

4.2 Complexity estimation for decision table hierarchy

Using DFC, the overall complexity of decision tables in the concept hierarchy is the sum of θ_{DFC} for each decision table. Similarly, for DTIC, the complexity estimation is again 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.

For example, consider the two concept hierarchies from Figure 1. Their overall complexities as measured by DFC are 15 and 21, respectively, and 20.76 bits and 27.68 bits as measured by DTIC. These measures confirm that the first decomposition is less complex and thus preferred to the second one. The original undecomposed decision table had DFC equal to 18 and DTIC equal to 28.53 bits. Therefore, in terms of DTIC both decompositions reduced the complexity, while using DFC this happened only with the first one.

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 consider 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

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 hierarchical structure itself. We call this measure *structure information content* (SIC). The more informative the hierarchy, the overall less complex the resulting decision tables.

For the two decompositions in Figure 1, the corresponding structure information contents are 7.77 bits and 0.85 bits, respectively. Since the first SIC is considerably greater than the second one, the first structure is more informative and its decision tables more compact.

5 Experimental evaluation

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

Each of six domains is represented with the initial decision table containing instances that completely cover the attribute space. Although the experiments could as well be done with sparser decision tables (see [25]), we wanted to focus in this article only on the discovery of concept hierarchies. Note that the proposed partition selection measures depend only on cardinalities of attributes and concepts, and not on the actual number of instances in decision tables. Furthermore, we have shown in [26] that by increasing the problem space coverage by training instances, the discovered concepts converge to those from complete training sets.

The results of decompositions are shown as concept hierarchy structures, where, unless otherwise noted, the labels of intermediate concepts indicate the order in which they were discovered.

5.1 Artificial domains

Three artificial domains were investigated:

1. a Boolean function
   $y = (x_1 \text{ OR } x_2) \text{ AND } x_3 \text{ AND } (x_4 \text{ XOR } x_5)$,
2. a six-attribute palindrome function,
3. a three-valued function
   $y = \mathrm{MIN}(x_1, \mathrm{AVG}(x_2, \mathrm{MAX}(x_3, x_4), x_5))$.

For the first function, the initial decision table has $2^5 = 32$ instances, $\theta_{\mathrm{DFC}} = 32$ and $\theta_{\mathrm{DTIC}} = 32$ bits. While decomposition with DTIC and CM discovered the anticipated hierarchy, the DFC-guided decomposition terminated too soon because the complexity condition did not allow to decompose the decision tables any further (see Figure 4). Note that the overall DFC is the same for all discovered hierarchies, while the structure information content is higher for those discovered by DTIC and CM. The decision tables (not

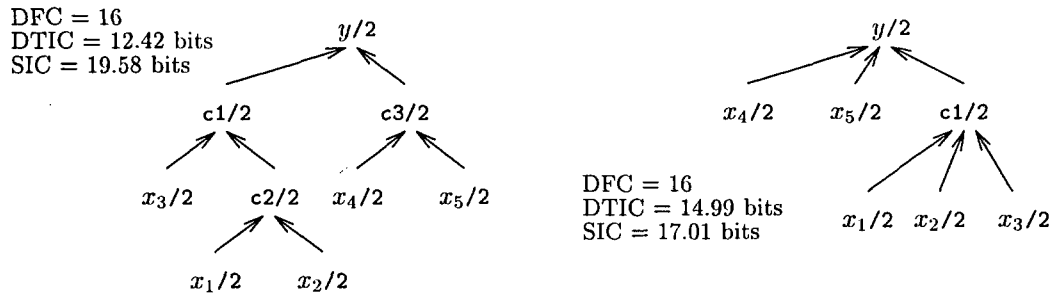

Figure 4: Decomposition of decision table representing the function $y = (x_1$ OR $x_2)$ AND x_3 AND $(x_4$ XOR $x_5)$ guided by DTIC and CM (left), and DFC (right).

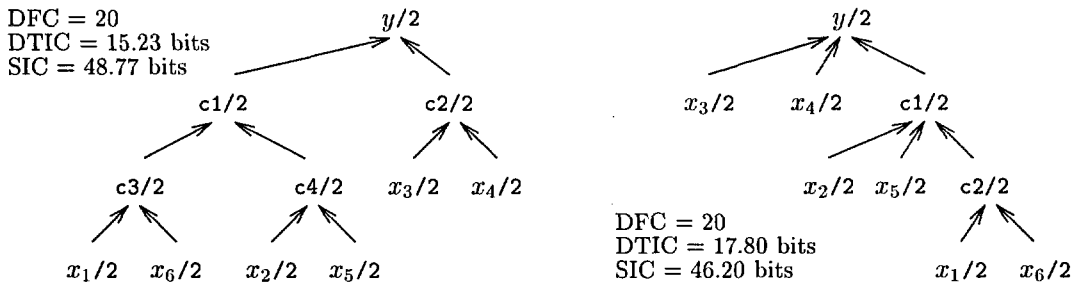

Figure 5: Decomposition of decision table representing the palindrome function guided by DTIC and CM (left), and DFC (right).

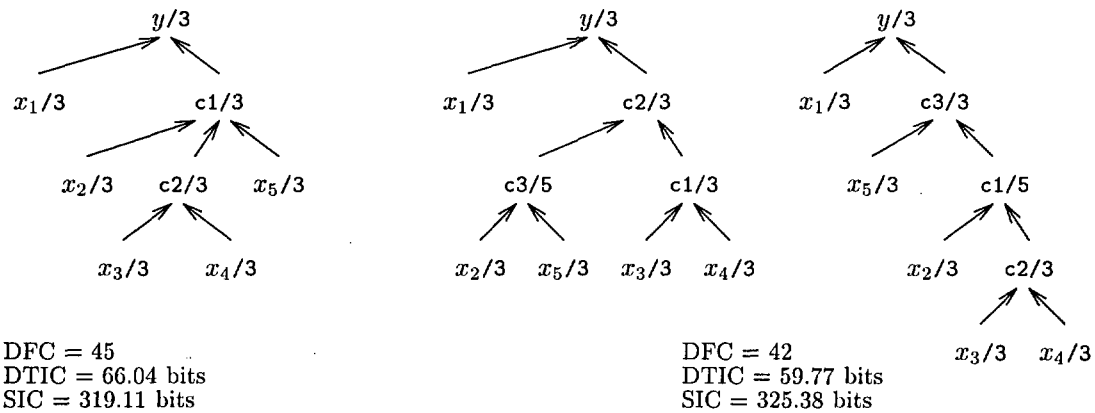

Figure 6: Decompositions of the function $y = \mathrm{MIN}(x_1, \mathrm{AVG}(x_2, \mathrm{MAX}(x_3, x_4), x_5))$: the anticipated hierarchy (left), the hierarchy discovered using CM (middle), and DFC and DTIC (right). The complexity and information measures for the latter two decompositions are the same.

shown in the figure) were checked for interpretability and were found to represent the expected functions.

The second function $y = \mathrm{PAL}(x_1, x_2, \ldots, x_6)$ 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, six Boolean attributes $x_1 \ldots x_6$ were used. The initial decision table has $\theta_{\mathrm{DFC}} = 64$ and $\theta_{\mathrm{DTIC}} = 64$ 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 hierarchy if the corresponding complexity condition would be changed to $\theta_{\mathrm{DFC}}(F) \geq \theta_{\mathrm{DFC}}(G) + \theta_{\mathrm{DFC}}(H)$. 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 concept hierarchy.

The third function $y = \mathrm{MIN}(x_1, \mathrm{AVG}(x_2, \mathrm{MAX}(x_3, x_4), x_5))$ uses ordinal attributes $x_1 \ldots x_5$ that can take the values 1, 2, and 3. While MIN and MAX have the standard interpretation, AVG computes the average of its arguments and rounds it to the closest integer. The initial decision table has $\theta_{\mathrm{DFC}} = 243$ and $\theta_{\mathrm{DTIC}} = 385.15$ bits. The anticipated and discovered hierarchies are shown in Figure 6. Quite surprisingly, in all three cases the decomposition yields a hierarchy with a higher structure information content than expected by introducing an additional five-valued intermediate concept. If this were removed, the discovered hirarchy and decision tables would have been the same as anticipated. It is also interesting to note that the hierarchy 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 concept hierarchies with regard to complexity.

5.2 DEX models

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 real-world decision making problems. DEX uses categorical attributes and expects the concept structure and corresponding decision tables to be defined by the expert. The formalism used to describe the DEX model and its interpretation are essentially the same as with concept hierarchies studied in this article. This makes decision models developed by DEX ideal benchmarks for the evaluation of decision table decomposition. In this article, 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 to rank applications for nursery schools [17]. 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 CAR and NURSERY datasets are available from the UCI Machine Learning Repository [16].

The goal of this experiment was to reconstruct these DEX models from examples. The learning instances were derived from the original models, where for all combinations of input attributes the class was determined by the corresponding model. The examples were stated as attribute-value vectors, hiding from the decomposition method any underlying conceptual structure of the domain.

The discovered hierarchies are given in Figures 7, 8, and 9. In all cases, the decomposition guided by DFC, DTIC, and CM 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 using CM. 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 concept hierarchies 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 hierarchies 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 concept hierarchies as the original ones but additionally decomposed the decision tables for `comfort` (CAR), `employ` (EMPLOY), and `struct+finan` (NURSERY). In this way it obtained less complex decision tables.

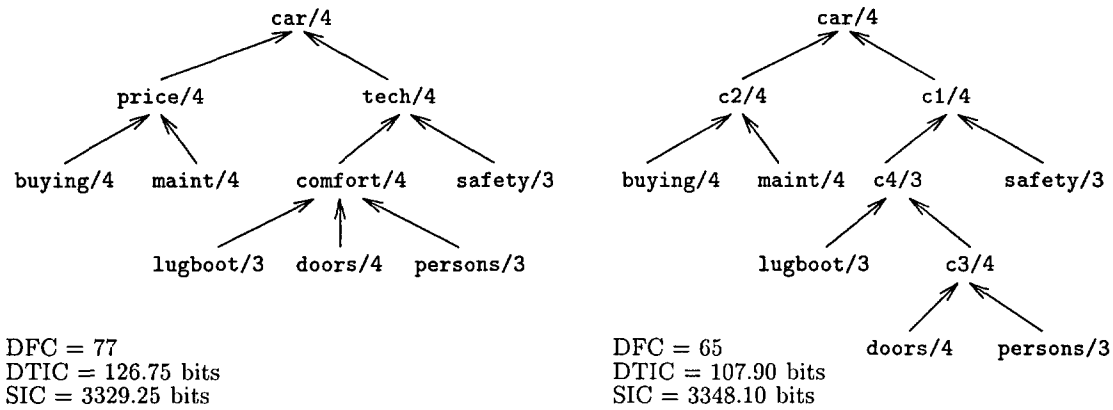

Figure 7: The original concept hierarchy of CAR (left) and the decompositions based on CM, DFC and DTIC (right).

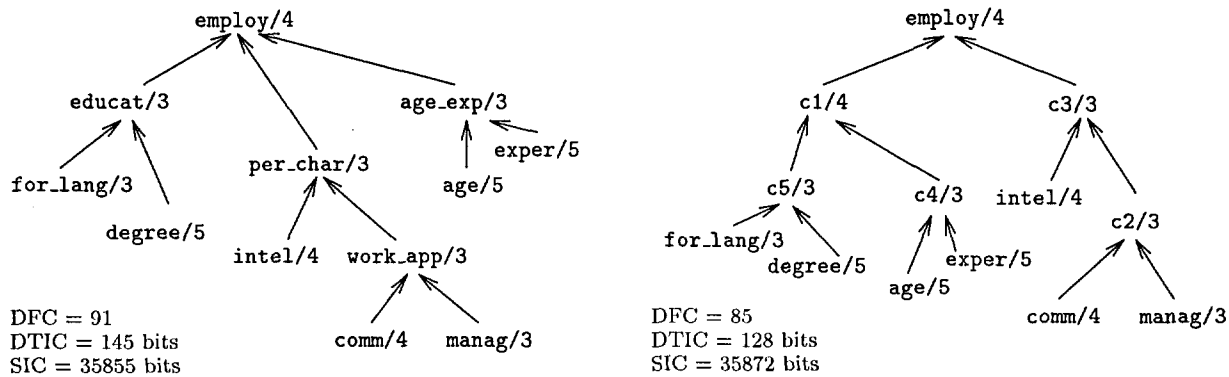

Figure 8: The original concept hierarchy of EMPLOY (left) compared to the hierarchy discovered by CM, DFC, and DTIC-guided decomposition (right).

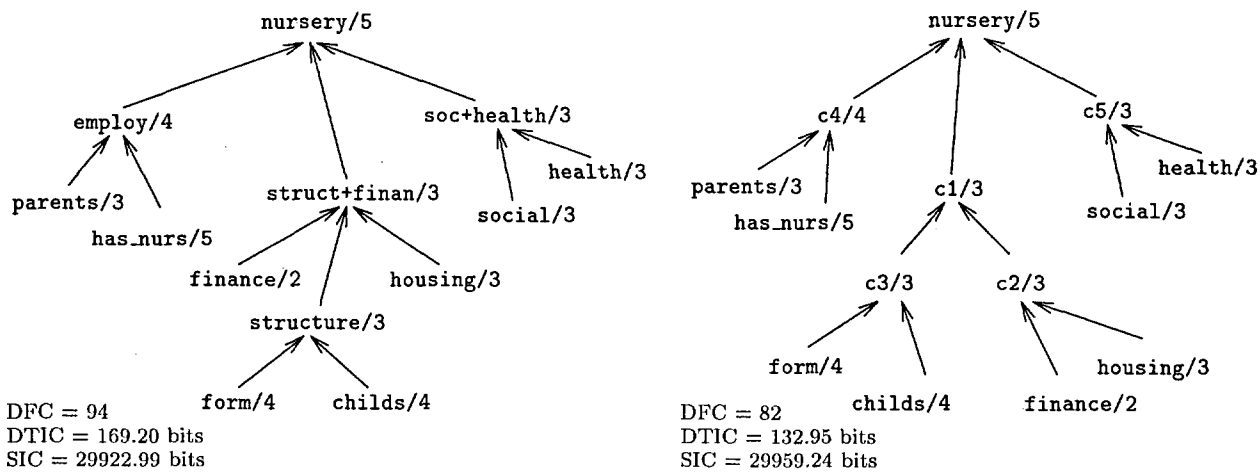

Figure 9: The original (left) and discovered concept hierarchy using CM, DFC and DTIC criteria (right) for NURSERY.

The derived decision tables were compared to the original ones and found to be the same but in the names used for instance labels (the decomposition uses abstract labels while the original decision tables use meaningful names). The only exception are decision tables for `tech` and `comfort` in the CAR domain, where the decomposition succeeded to find a more compact representation.

6 Conclusion

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

The experimental evaluation exposed the deficiency of DFC when decomposing a decision table that expresses a Boolean function. This may be alleviated by relaxing the DFC complexity condition. In more complex domains with multi-valued attributes, the decomposition guided by any of the proposed criteria discovered concept hierarchies that were very similar to those expected. Furthermore, the discovered hierarchies were equal to or even better than the anticipated ones in terms of the complexity of decision tables and structure information content. 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 hierarchies 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 a 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 circuit design. However, experiments in non-trivial domains like DEX's strongly encourage further research and development of this method for machine learning and knowledge discovery. As the method has recently been extended to deal with continuous attributes [9] and noise [25], further research is needed to assess the quality of corresponding partition selection criteria under these extensions.

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] M. Bohanec, B. Zupan, I. Bratko, and B. Cestnik. A function decomposition method for development of hierarchical multi-attribute decision models. In *Proc. 4th Conference of the International Society for Decision Support Systems (ISDSS-97)*, pages 503–514, Lausanne, Switzerland, July 1997.

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

[9] J. Demšar, B. Zupan, M. Bohanec, and I. Bratko. Constructing intermediate concepts by decomposition of real functions. In M. van Someren and G. Widmer, editors, *Proc. European Conference on Machine Learning, ECML-97*, pages 93–107, Prague, April 1997. Springer.

[10] C. Files, R. Drechsler, and M. Perkowski. Functional decomposition of MVL functions using multi-valued decision diagrams. In *International Symposium on Multi-Valued Logic*, may 1997.

[11] J. A. Goldman. Pattern theoretic knowledge discovery. In *Proc. the Sixth Int'l IEEE Conference on Tools with AI*, 1994.

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

[13] 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.

[14] 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.

[15] 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.

[16] P. M. Murphy and D. W. Aha. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/mlrepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1994.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

[27] B. Zupan, M. Bohanec, J. Demšar, and I. Bratko. Feature transformation by function decomposition. *IEEE Intelligent Systems & Their Applications*, 13(2):38–43, March/April 1998.