Algorithms for developing complex rules in DEX methodology: Improvement and performance analysis

Adem Kikaj[1,2], **Marko Bohanec**[2]
[1]*Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia*
adem.kikaj@ijs.si
[2]*Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia*

Abstract: DEX (Decision EXpert) is a qualitative multi-criteria decision-making methodology. The models in DEX are developed based on attributes, scales of attributes, the tree of attributes and decision rules. DEXi is a computer program which implements the DEX method. The decision rules in DEXi are represented in a table, where each row of this table is called an elementary decision rule. In order to represent decision rules in a more compact way, DEXi uses an algorithm for developing complex decision rules. This paper formulates the complex rules problem, presents the existing complex rules algorithm and proposes a new algorithm, aimed at improving the efficiency.

Keywords: algorithms, DEX method, decision rules, complex rules, multiple criteria analysis

1 Introduction

Decision making is a difficult and complex process. In this process, a decision maker (DM) faces several decision alternatives. To choose a particular alternative from the set of possible alternatives a decision-analyze approach (Bouyssou et al., 2006; Greco et al., 2005) can help to satisfy the aims or goals of a decision maker. Decision analysis (Bouyssou et al., 2006; Greco et al., 2005) is the discipline used to help a decision maker to deal with uncertainty, complexity, risk, and trade-offs of the decision. The idea of decision analysis is to build a decision model which can help decision makers to evaluate alternatives and to choose the best action.

Since a Decision Maker in a decision problem have to deal with multiple and possibly conflicting criteria the Multiple Criteria Decision Analysis (MCDA) or Multiple Criteria Decision Making (MCDM) (Bouyssou et al., 2006), provides methods used to structure, plan and solve such problems.

In this work, we focus on one MCDM method, called DEX. DEX is a qualitative multi-criteria decision making methodology with rule-based expert systems (Trdin & Bohanec, 2018; Bohanec, 2017; Bohanec et al., 2013). The method supports decision makers in making a complex decision based on multiple, possibly conflicting, attributes. DEXi (Bohanec, 2008) is a computer program for multi-attribute decision making which implements the DEX method. DEXi has been used in different areas such as Information technology, projects, companies, personnel

management, medicine and health-care and many others areas. DEXi models are developed by defining attributes, scales, the tree of attributes and decision rules. Attributes are variables which determine the properties of a specific object in the model. In DEXi attributes can be either basic attributes or aggregate attributes. Aggregate attributes have subordinate attributes and decision rules. Tree of attributes is organized attributes hierarchically into a tree. Scale represents a set of values that can be assigned to an attribute. DEXi uses qualitative and discrete values of scales such as: 'excellent', 'acceptable', 'inappropriate', etc. Decision rules define the aggregation aspect of option evaluation. In DEXi, a decision rule table is created from aggregated attributes by mapping all the combinations between subordinate attributes of the aggregated attribute. In order to present the decision rule table in a more compact way, DEXi uses complex rules.

In this work we formulate the complex rules problem, present the existing complex rules and the proposed algorithm. Then, we analyze and compare the two algorithms. The structure of this paper is as follows. Section 2 formulates the complex rules problem. Section 3 describes the DEXi's algorithm used to generate complex rules. Section 4 presents the proposed algorithm. Comparison of the algorithm and results of complex rules are discussed in section 5. Section 6 concludes the work.

2 Complex Rules

In DEXi decision table represents an aggregation of subordinate attributes to the attribute. This decision table is a finite set of decision rules also called as elementary decision rules. Elementary decision rules are generated from the attribute values of the specific aggregated attribute by following a specific function. In DEXi X is a set of attributes:

$$X = \{x_1, x_2, ..., x_n\}. \tag{1}$$

where each attribute $x_i \in X$ denotes a single attribute, either basic or aggregate which have a scale $D_i \in D$. The scale consists of a set of qualitative values that can be assigned to the attributes:

$$D_i = \{w_{i_1}, w_{i_2}, ..., w_{i_{m_i}}\}. \tag{2}$$

Here, m_i is the number of values in the scale of attribute x_i.

The condition parts of decision rules for the aggregated attribute $x_i \in X$ are defined by the Cartesian product of its subordinates attributes scales D.

$$C : D_{x_1} \times D_{x_2} \times ... \times D_{x_n} \tag{3}$$

All the decision rules in order to be evaluated needs a function. We denote $f_i \in F$ as the aggregation function corresponding to the aggregated attribute X. Each aggregation function f_i is a total function that maps all scale value combinations of x_i's scale values to an interval scale value D_i:

$$f_i : D_{x_{1_i}} \times D_{x_{2_i}} \times ... \times D_{x_{n_i}} \rightarrow I(D_i) \tag{4}$$

where,

$$I(D_i) = [v_l, v_h] = \{v_j \mid v_l \leq v_j \leq v_h \wedge v_l, v_j, v_h \in D_i\} \tag{5}$$

In follow we represents a single elementary rule:

$$er : \underbrace{< D_{x_{1_i}}, D_{x_{2_i}}, ..., D_{x_{n_i}} >}_{\text{condition part}} \rightarrow \underbrace{I(D_i)}_{\text{interval of decision value}} \tag{6}$$

In order to present all generated elementary rules in a more compact and comprehensible way, DEXi uses complex rules algorithm to join several elementary rules Eq. 6 with the same decision value $w_i \in D_{aggregatedAttribute}$.

A complex rule consists of the condition and decision value part. The conditional part is a conjunction of clauses, and each CLAUSE can represent an interval, while the decision value compared with the elementary rule Eq. 6 is not an interval but a single scale value w_i of the set $D_{aggregatedAttribute}$. In follow is represented a single complex rule:

$$cr : \underbrace{< I(D_{x_1}), I(D_{x_2}), ..., I(D_{x_n}) >}_{\text{cover area}} \rightarrow \underbrace{w_i}_{\text{single decision value}} \tag{7}$$

Based on the cover area of complex rules an interval can be denoted as:

- '*': the asterisk include all possible scale values of a specific attributes $x_i \in X$, e.g. set of D_i, see expression 2
- '$>= w_{i_1}$': stands for *better than or equal to value*, $w_{i_1} \in D_i$ (2)
- '$<= w_{i_1}$': stands for *worse than or equal to value*, $w_{i_1} \in D_i$ (2)
- '$w_{i_1} : w_{i_2}$': interval between value w_{i_1} and value w_{i_2} including the two values, $w_{i_1}, w_{i_2} \in D_i$ (2)

For a better presentation of elementary and complex rules in this work we are going to use one of the DEXi models known as *CAR Evaluation model* (Bohanec, 2008) see Fig 1.

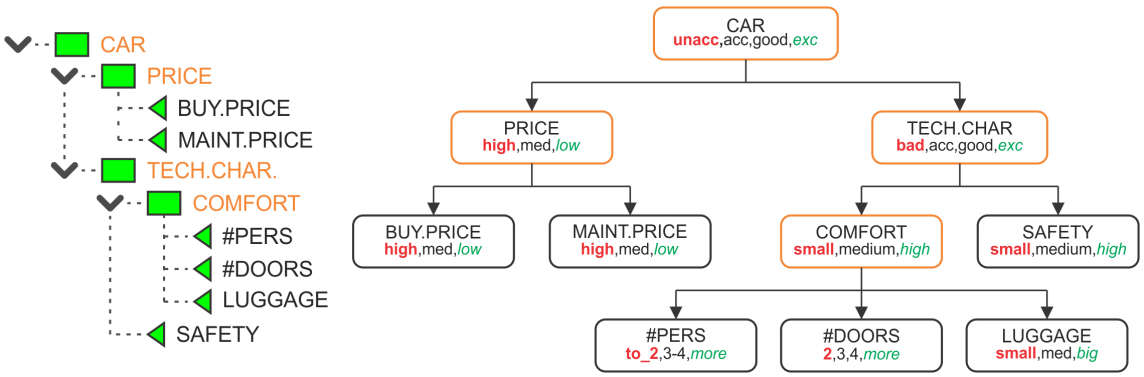


Figure 1: Tree structure of Car Evaluation model. Names and values of attributes are shown for each attribute. The attributes colored orange are aggregated attributes

In the Car Evaluation Model we have attributes:

$$X = \{CAR, PRICE, TECH.CHAR, BUY.PRICE, MAINT.PRICE, \\ COMFORT, SAFETY, \#PERS, \#DOORS, LUGGAGE\} \tag{8}$$

In Car Evaluation Model model, Fig. 1, all the scales of attributes are ordered preferentially for instance:

$$D_{CAR} = \{unacc, acc, good, vgood\} \tag{9}$$

Qualitative scales in Eq. 9 can be mapped also with numbers by getting ordinal scales, e.g. $O(unacc) = 0$. Since complex rules are areas of scales interval we will convert the qualitative scales to ordinale ones as follow:

$$O_{CAR} = \{0, 1, 2, 3\} \tag{10}$$

By applying Cartesian product Eq. 3 to subordinate attributes of the aggregated attribute CAR, Fig. 1, we get the following conditional parts of the elementary rules.

$$CAR = PRICE \times TECH = \{(high, bad), (high, acc), (high, good), (high, exc), \\ (medium, bad), (medium, acc), (medium, good), (medium, exc), \\ (low, bad), (low, acc), (low, good), (low, exc)\} \tag{11}$$

After applying Eq. 11 to the aggregated attribute and evaluating them with decision values we have table 1 with qualitative values of attributes together with decision qualitative values of CAR aggregated attribute.

Since we have to deal with two subordinate attributes of CAR aggregated attribute we present the elementary rules in a matrix together with the decision values see Fig. 2

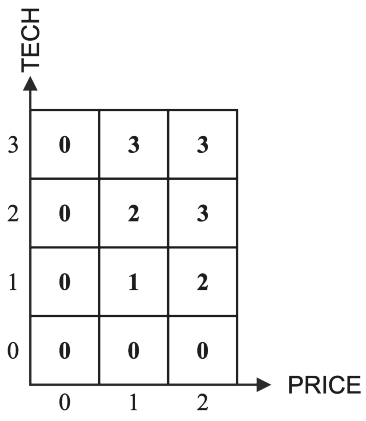


Figure 2: Elementary rules of CAR aggregated attribute presented in matrix

PRICE	TECH	**CAR**
high	bad	**unacc**
high	acc	**unacc**
high	good	**unacc**
high	exc	**unacc**
medium	bad	**unacc**
medium	acc	**acc**
medium	good	**good**
medium	exc	**exc**
low	bad	**unacc**
low	acc	**good**
low	good	**exc**
low	exc	**exc**

(a) Elementary rules with qualitative values

PRICE	TECH	**CAR**
0	0	**0**
0	1	**0**
0	2	**0**
0	3	**0**
1	0	**0**
1	1	**1**
1	2	**2**
1	3	**3**
2	0	**0**
2	1	**2**
2	2	**3**
2	3	**3**

(b) Elementary rules mapped with numbers

Table 1: Elementary rules of CAR aggregated attribute

3 DEXi Complex Rule Algorithm

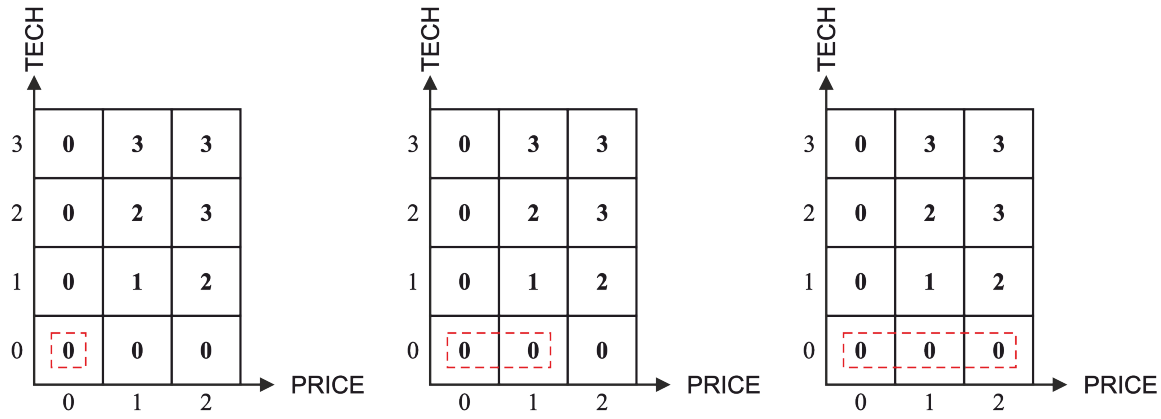
DEXi complex rule algorithm generates complex rules 7 by finding areas limited by bounds which may cover more than one elementary rule 6.

The DEXi complex rule algorithm tries to extend two bounds, respectively low and high bound Fig. 3 which means it tries to extend the cover area of the complex rule 7. These bounds are extended in the different direction for each subordinate attributes. Complex rules are generated by covering elementary rules with same decision value $w_i \in D_i$. When bounds reach their limits respectively the last qualitative value of its set the $w_{i_{n_i}}$ value Eq. 2 or when bounds cover elementary rules with different decision value $w_i \in D_i$, one complex rule is generated.

In Fig. 3 is shown the process of generating complex rules by increasing the high bound or increasing the $I(D_{PRICE})$ in $PRICE$ direction with decision value $w = low$ or $O_{CAR} = 0$.

In Fig. 3 one complex rule is generated. The process started from the low and high bound with the lowest value for the group of elementary rules with decision value $O_{CAR} = 0$. This process stopped when the high bound reached the highest limit in $PRICE$ direction. The generated complex rule now covers elementary rules from the Low bound $Low = 00$ to the high bound $High = 20$ or $I(O_{PRICE}) = [0, 2]$. Since it covers all possible finite scale values of the $PRICE$ attribute we write it with the asterisk '*' and for the $TECH$ attribute, it covers only one discrete value which is 0 or $I(O_{TECH}) = [0, 0]$. The generated complex rule is shown in the Table 2. This complex rule based on Eq. 7 takes form as follow:

$$cr :< I(D_{PRICE}), I(D_{TECH}) >\rightarrow 0, I(D_{PRICE}) = [0, 2] \wedge I(D_{TECH}) = [0, 0]$$



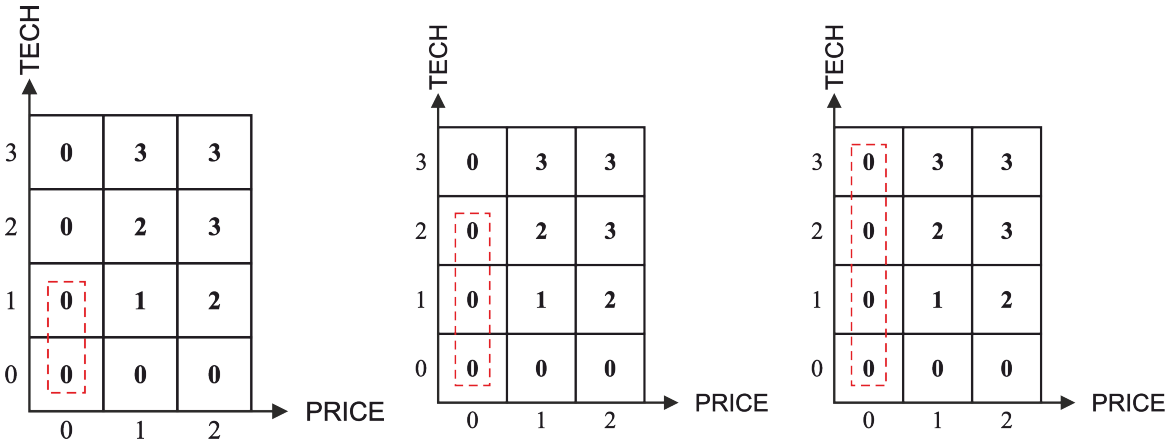
(a) $Low = 00, High = 00$ (b) $Low = 00, High = 10$ (c) $Low = 00, High = 20$

Figure 3: DEXi complex rules algorithm extending high bound in the *PRICE* direction

PRICE	TECH	**CAR**
*	bad	**unacc**

Table 2: Complex rules of CAR aggregated attribute

After the limit of the *PRICE* attribute is reached the algorithm tries to extend bound in the *TECH* attribute direction. It fails to extend in the *TECH* attribute direction since there are different decision values such for $High = 21$ is 2 and $High = 11$ is 1. When it comes to the starting point it extends the High bound as is shown in Fig. 4.



(a) $Low = 00, High = 01$ (b) $Low = 00, High = 02$ (c) $Low = 00, High = 03$

Figure 4: DEXi complex rules algorithm extending high bound in *TECH* direction

After covering elementary rules in the *TECH* attribute direction we have two complex rules shown in Table 3.

PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**

Table 3: Complex rules of CAR aggregated attribute

At this point, we have covered all elementary rules with decision value $w = low$ or $O_{CAR} = 0$ for the CAR aggregated attribute.

The complex rule algorithm continues the process of finding complex rules with decision value $O_{CAR} = 1$ since it has covered all elementary rules with decision value $O_{CAR} = 0$. There is just one elementary rule with decision value $O_{CAR} = 1$ and this remains same as a complex rule and it is generated as it is in Table 4.

PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**
medium	acc	**acc**

Table 4: Complex rules of CAR aggregated attribute

For the decision value $O_{CAR} = 2$ there is also no possible way to extend it in any direction of one of the subordinate attributes Fig. 5. In this case, two elementary rules are also treated as complex rules.

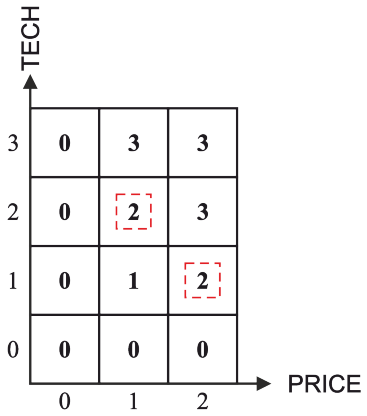


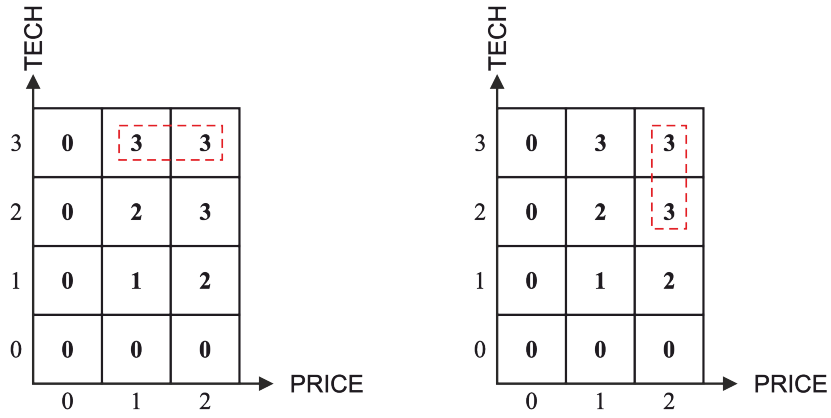
Figure 5: Elementary rules of CAR aggregated attribute with decision value $O_{CAR} = 2$

PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**
medium	acc	**acc**
medium	good	**good**
low	acc	**good**

Table 5: Complex rules of CAR aggregated attribute

The two elementary rules with decision value $O_{CAR} = 2$ will be added in complex rules output as complex rules Table 5.

Also, after complex rules algorithm finishes covering elementary rules with decision value $O_{CAR} = 2$ it tries to cover elementary rules with the next decision value which is $O_{CAR} = 3$. In this case, the complex rules algorithm generates two complex rules Fig. 6.



(a) $Low = 13, High = 23$ (b) $Low = 22, High = 23$

Figure 6: Complex rules algorithm with decision value $O_{CAR} = 3$

After the two complex rules are generated, the complex rules algorithm terminates since it has covered all elementary rules of CAR aggregated attribute. The final complex rules are in Table 6.

PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**
medium	acc	**acc**
medium	good	**good**
low	acc	**good**
>=medium	exc	**exc**
low	>=good	**exc**

Table 6: Generated complex rules for *CAR* aggregated attribute

Complex rules shown in table 6 covers all elementary rules of *CAR* aggregated attribute shown in Table 1. The 12 elementary rules are represented by 7 complex rules. The generated complex rules are a compact presentation of elementary rules and this is done by using interval values such as '*' and '>='.

The complex rule algorithm begins with a precisely specified input which is the low and high bound of a group of elementary rules with the same decision value. This algorithm terminates in a finite time and on termination the algorithm returns the complex rules. The pseudo-code of the DEXi complex rule algorithm is shown in Algorithm 1.

Algorithm 1 DEXi's complex rule algorithm
INPUT: $l :< D_{x1_1}, D_{x2_1}, ..., D_{xn_1} >$, $h :< D_{x1_1}, D_{x2_1}, ..., D_{xn_1} >$, $t = w_i, w_i \in D(CAR)$, $max = l :< D_{x1_{m_i}}, D_{x2_{m_i}}, ..., D_{xn_{m_i}} >$
OUTPUT: $CR = \{c_1, c_2, ..., c_n : c \in cr\}$

```
1:  procedure ComplexRule(l, h, t, max)
2:      for i ← 0 to |f_i| do
3:          er ← f_i
4:          for j ← 0 to |C| do
5:              if er[j] ≥ l[j] and er[j] ≤ h[j] then
6:                  cover ← TRUE
7:              end if
8:          end for
9:          if cover then
10:             cover ← I(D_er) == t
11:         end if
12:     end for
13:     if cover then
14:         for i ← 0 to |h| do
15:             if h[i] < max[i] then
16:                 h[i]++
17:                 ComplexRule(l, h, t, max)
18:             end if
19:         end for
20:         for i ← 0 to |l| do
21:             if l[i] > 0 then
22:                 l[i]−−
23:                 ComplexRule(l, h, t, max)
24:             end if
25:         end for
26:         complexRules.add(l+h)
27:     end if
28: end procedure
```

4 The proposed algorithm

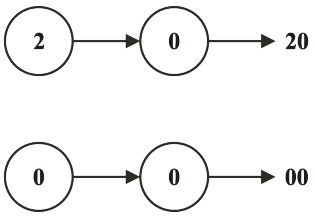
The motivation of this work to propose a new algorithm which provides complex rules is based on optimization of the algorithm complexity. Since in the section 5 we have shown that the DEXi complex rules algorithm have exponential complexity we propose a new algorithm aimed to improve in the manner of complexity. Also, the motivation is based on the efficiency of provided results respectively the number of complex rules. The proposed algorithm uses a different covering logic compared to the DEXi complex rules algorithm shown in this section.

In order to describe the proposed algorithm, we will work again with the same elementary rules from the CAR aggregated attribute shown in Table 1. From the

grouped elementary rules with the same decision value, the algorithm takes the lowest possible scale value for each subordinate attribute and uses it as the lowest possible bound $< D_{x_{1_1}}, D_{x_{2_1}}, ..., D_{x_{n_1}} >$ then it tries to count to the highest bound $< D_{x_{1_{m_i}}}, D_{x_{2_{m_i}}}, ..., D_{x_{n_{m_i}}} >$. In the counting process if all elementary rules are found then one complex rule is generated from the lowest to the highest bound.

PRICE	TECH	**CAR**
0	0	**0**
0	1	**0**
0	2	**0**
0	3	**0**
1	0	**0**
2	0	**0**

(a) Elementary rules with decision value $O_{CAR} = 0$



(b) $Low = 00, High = 20$

Table 7: Elementary rules with decision value $O_{CAR} = 0$ and created bounds from the proposed algorithm

The lowest created bound for the elementary rules with decision value $O_{CAR} = 0$ is the elementary rule $< 00 >$ which means the lowest possible value for the *PRICE* attribute is $D_{PRICE} = 0$ and also for the *TECH* attribute is $D_{TECH} = 0$. The proposed algorithm counts from $< 00 >$ to the last elementary rule from the Table 7a, which is $< 20 >$. In this example these two bounds successfully find elementary rules in between and the generated complex rule is shown in Table 8.

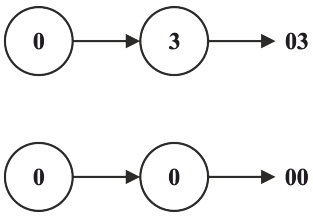
PRICE	TECH	**CAR**
*	bad	**unacc**

Table 8: Generated complex rules from the proposed algorithm for *CAR* aggregated attribute

Since there are still elementary rules that have not been covered the algorithm finds the next highest bound in the elementary rules which is $< 03 >$ and tries again from the same lowest bound to reach the highest one.

PRICE	TECH	**CAR**
0	1	**0**
0	2	**0**
0	3	**0**

(a) Elementary rules not covered



(b) $Low = 00, High = 03$

Table 9: Elementary rules not covered with decision value $O_{CAR} = 0$ and created bounds from the proposed algorithm

In this case, the count is again possible from the lowest bound $< 00 >$ to the highest one $< 03 >$ and a new complex rule is generated, Table 10.

PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**

Table 10: Generated complex rules from the proposed algorithm for CAR aggregated attribute

The proposed algorithm continues to find complex rules with the next decision value $O_{CAR} = 1$. Since there is just one elementary rule with this decision value the same elementary rule is treated as the complex rule and the complex rules are shown in Table 11.

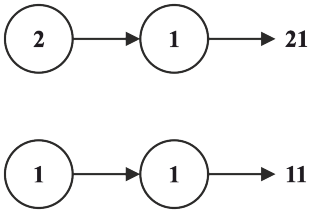
PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**
medium	acc	**acc**

Table 11: Generated complex rules from the proposed algorithm for CAR aggregated attribute

The next decision value is $O_{CAR} = 2$ and there are two elementary rules:

PRICE	TECH	**CAR**
1	2	**2**
2	1	**2**

(a) Elementary rules with decision value $O_{CAR} = 2$



(b) $Low = 11, High = 21$

Table 12: Elementary rules with decision value $O_{CAR} = 2$ and created bounds from the proposed algorithm

The lowest bound generated in this case is $< 11 >$ where the first value is the lowest possible value $D_{PRICE} = 1$ for the *PRICE* attribute and for the second value is the lowest possible value$D_{TECH} = 1$ for the *TECH* attribute from the elementary rules in Table 12a. In this case, the proposed algorithm can not count because there is no elementary rule with value $< 11 >$ in Table 12a. The proposed algorithm changes the lowest bound according to the highest one.

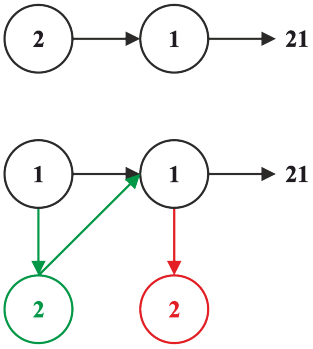


Figure 7: The new low bound generated according to the high bound

The new low bound is $< 21 >$, Fig. 7, the proposed algorithm tries from the right to the left of attributes to increase the low bound value. The reason why it goes from the right to the left is that in that order it can be the next lowest possible bound. In our case, the proposed algorithm tries to increase the lowest bound to $< 12 >$ but since the second value respectively the *TECH* attribute value in the highest bound is $D_{TECH} = 1$ it can not increase to 2. The proposed algorithm tries to increase the next attribute which is the *PRICE* attribute value since this is possible to increase to $D_{PRICE} = 2$ the new lowest bound is $< 21 >$.

From this point the proposed algorithm does not count or find the elementary rules between the low and high bound since there are equal. The new generated complex rule is $< 21 >$ and the complex rules are shown in Table 13.

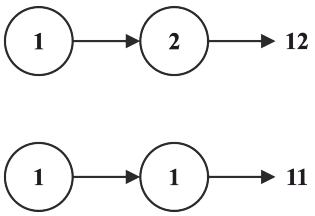
PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**
medium	acc	**acc**
low	acc	**good**

Table 13: Generated complex rules from the proposed algorithm for CAR aggregated attribute

Since there is one more elementary rule not covered the proposed algorithm tries to cover it from the low bound as at the beginning with value $< 11 >$ and tries to reach the highest one which is $< 12 >$.

PRICE	TECH	**CAR**
1	2	**2**

(a) Elementary rules not covered with decision value $O_{CAR} = 2$



(b) $Low = 11, High = 12$

Table 14: Elementary rules not covered with decision value $O_{CAR} = 2$ and created bounds from the proposed algorithm

In this case, the proposed algorithm can not find the rules between generated bounds, so it creates a new low bound Fig 8.

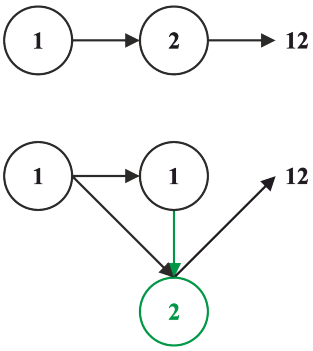


Figure 8: The new low bound generated according to the high bound

Again, in this case, the generated low bound and the high bound are equal and this elementary rule is treated as the complex rule. The complex rules are shown in Table 15.

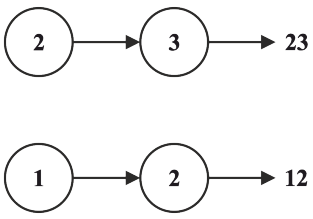
PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**
medium	acc	**acc**
low	acc	**good**
medium	good	**good**

Table 15: Generated complex rules from the proposed algorithm for CAR aggregated attribute

After all the elementary rules with decision value $O_{CAR} = 2$ are covered then the proposed algorithm tries to create complex rules from the elementary rules with the next decision value $O_{CAR} = 3$, in total there are three elementary rules:

PRICE	TECH	**CAR**
1	3	**3**
2	2	**3**
2	3	**3**

(a) Elementary rules with decision value $O_{CAR} = 3$



(b) $Low = 12, High = 23$

Table 16: Elementary rules with decision value $O_{CAR} = 3$ and created bounds from the proposed algorithm

The proposed algorithm cannot find the elementary rule $< 12 >$ so it changes the low bound according to the highest bound $< 23 >$, Fig. 9.

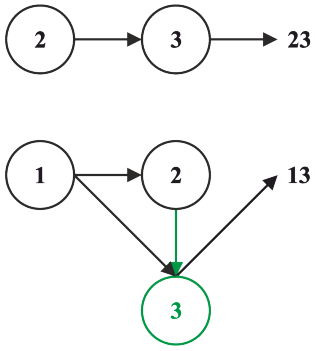


Figure 9: The new low bound generated according to the high bound

Now the proposed algorithm can count between two bounds and it generates a new complex rule. The complex rules are shown in Table 17.

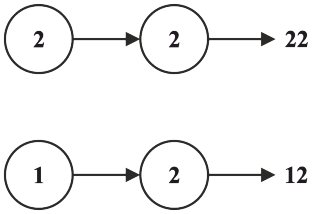
PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**
medium	acc	**acc**
low	acc	**good**
medium	good	**good**
>=medium	exc	**exc**

Table 17: Generated complex rules from the proposed algorithm for *CAR* aggregated attribute

Again, there are elementary rules left without being covered so there is new high bound and the low bound is again the same one generated at the beginning which is $< 12 >$.

PRICE	TECH	**CAR**
2	2	**3**

(a) Elementary rules not covered with decision value $O_{CAR} = 3$



(b) $Low = 12, High = 22$

Table 18: Elementary rules not covered with decision value $O_{CAR} = 3$ and created bounds from the proposed algorithm

In this case, a new low bound is generated Fig. 10, because there is no elementary rule same as the low bound 18b.

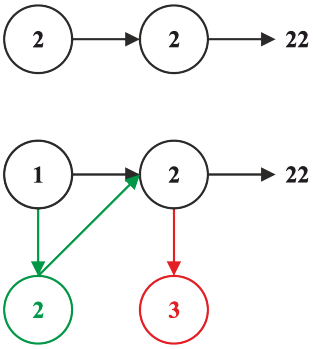


Figure 10: The new low bound generated according to the high bound

In this case, the two bounds are equal and the elementary rule is treated as the new complex rule. All the elementary rules now have been covered and the final complex rules table is:

PRICE	TECH	**CAR**
*	bad	**unacc**
high	*	**unacc**
medium	acc	**acc**
low	acc	**good**
medium	good	**good**
>=medium	exc	**exc**
low	good	**exc**

Table 19: Generated complex rules from the proposed algorithm for *CAR* aggregated attribute

The pseudo-code of the proposed algorithm is shown in Algorithm 2.

Algorithm 2 The proposed algorithm
INPUT: gER = grouped elementary rules, low = array of low bounds for each decision value
OUTPUT: $CR = \{c_1, c_2, ..., c_n : c \in cr\}$

```
1:  procedure NewComplexRule()
2:      for i ← 0 to |gER| do
3:          for j ← 0 to |gER[i]| do
4:              er ← gER[i][j]
5:              if !er.covered() then
6:                  lowBound = low[i]
7:                  highBound = er
8:                  while !Find(lowBound, highBound) do
9:                      lowBound++
10:                 end while
11:                 complexRules.add(lowBound+highBound)
12:             end if
13:         end for
14:     end for
15: end procedure
```

5 Comparison and Results

The comparison between the DEXi complex rule algorithm and the proposed algorithm is based on time complexity, the results that provide, respectively the number of complex rules that these two algorithm generates, and the running time since both algorithms are implemented in Java based on the JDEXi library (*JDEXi:*

Open-source DEXi Java Library, 2018) and perform in the same hardware configuration.

To facilitate the comparison based on time complexity for the algorithms in this work we will denote by $C_{i,j}$ the subprogram that consists of lines i to j to calculate the costs for executing this subprogram.

We begin to calculate the time complexity of the DEXi complex rule algorithm, Algorithm 1. We start with the first block $C_{2,12}$. This is a block with some constants and the n, m are denoted for:

- n: number of elementary rules ($|f_i|$, C_2 of Algorithm 1)
- m: number of subordinate attributes ($|C|$, C_4 of Algorithm 1)

$$C_{2,12} = \sum_{i=1}^{n} \sum_{j=1}^{m} c = O(nm), n \neq m \tag{12}$$

After having the worst case for the subprogram $C_{2,12}$ we will calculate the recurrence for the DEXi complex rule algorithm $C_{14,19}$. This is also a block with some constants and here the m is denoted for:

- m: scale of specific subordinate attribute ($|h|$, C_{14} of Algorithm 1)

$$C_{14,19} = T(l, h) = mT(l, h + 1) + mc$$

with

$$T(l, m) = mc$$

By applying the recurrence several times

$$T(l, h) = mT(l, h + 1) + mc$$

$$T(l, h + 1) = mT(l, h + 2) + mc$$

$$T(l, h) = m[mT(l, h + 2) + mc] + mc$$

$$T(l, h) = m^2T(l, h + 2) + m^2c + mc$$

$$...$$

$$T(l, n) = m^nT(l, n + 1) + m^nc + ... + m^2c + m^1c$$

Let $n + 1 = m$

$$T(l, n) = m^nT(l, m) + m^nc + ... + m^2c + m^1c$$

$$T(l, n) = m^n(mc) + m^nc + ... + m^2c + m^1c$$

$$T(l, n) = m^{n+1}c + m^nc + ... + m^2c + m^1c$$

and finally

$$T(l, n) = mc(m^{n+1} + m^n + ... + m + 1) = mc \sum_{i=0}^{m} m^i$$

$$T(l,n) = mc\frac{m^n - 1}{m - 1} = O(m^n) \tag{13}$$

From the Eq. 13 we defined that the DEXi complex rule algorithm is an algorithm with exponential complexity.

For the proposed algorithm we will calculate the time complexity as one algorithm and not by subprograms as for the first algorithm. The proposed algorithm has some constants and the n, m and o are denoted for:

- n: number of grouped elementary rules ($|gER|$, C_2 of Algorithm 2)
- m: number of elementary rules for a group ($|gER[i]|$, C_3 of Algorithm 2)
- o: number of all elementary rules (C_8 of Algorithm 2)

$$C_{1,15} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{o} c = \Omega(nmo), n \neq m \neq o$$

The best case for the proposed algorithm is $\Omega(nmo)$. This can be achieved if the method *Find* C_8, return a positive result, respectively true. The method *Find* C_8 is a loop which visits all elementary rules without being grouped.

For the worst case, it means that we have to create new low bound and to visit all the elementary rules every time that we have new bounds. In this case, the *while* loop C_8 will generate new low bound till it can generate a correct complex rule. We will denote with p the while loop:

$$C_{1,15} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{o}\sum_{l=1}^{p} c = O(nmop), n \neq m \neq o \neq p$$

Regarding the comparison based on the number of complex rules that two algorithms generate we are going to use a DEXi example with three subordinate attributes and one aggregated attribute Fig. 11.

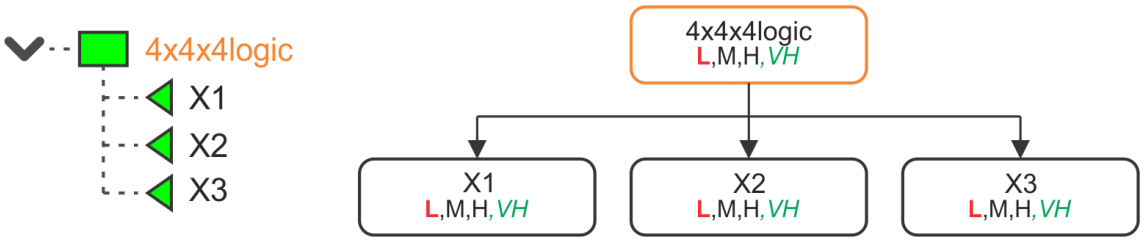


Figure 11: Tree structure of 4x4x4 Logic model. Names and values of attributes are shown for each attribute. The attribute with orange color is the aggregated attribute

The number of complex rules generated for the decision value $O_{4x4x4logic} = 0$ or $D_{4x4x4logic} = L$ is same for both algorithms, in total there are three complex rules generated Fig 12.

In this example both algorithms generates the same number of complex rules for the elementary rules with decision value $O_{4x4x4logic} = 0$ or $D_{4x4x4logic} = L$,

$O_{4x4x4logic} = 1$ or $D_{4x4x4logic} = M$ and $O_{4x4x4logic} = 2$ or $D_{4x4x4logic} = H$. The proposed algorithm generates less complex rules for elementary rules with decision value $O_{4x4x4logic} = 2$ or $D_{4x4x4logic} = H$ Fig. 13.

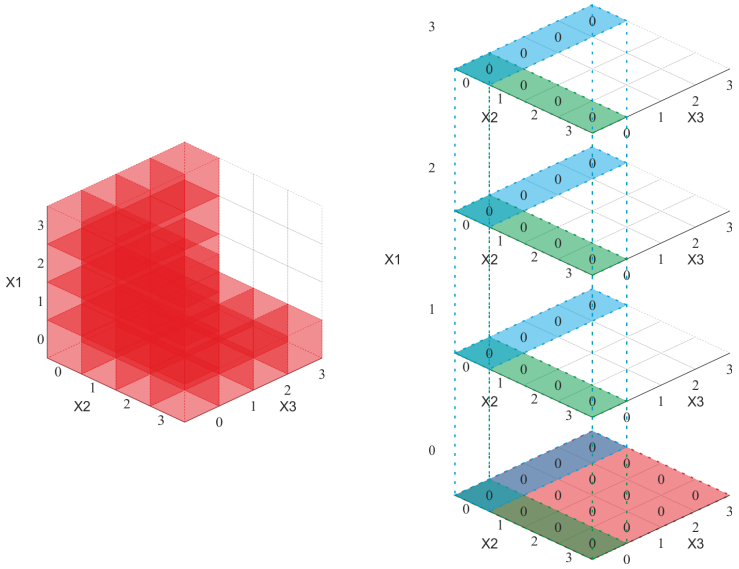


Figure 12: Elementary rules with decision value $O_{4x4x4logic} = 0$ and complex rules with different colors

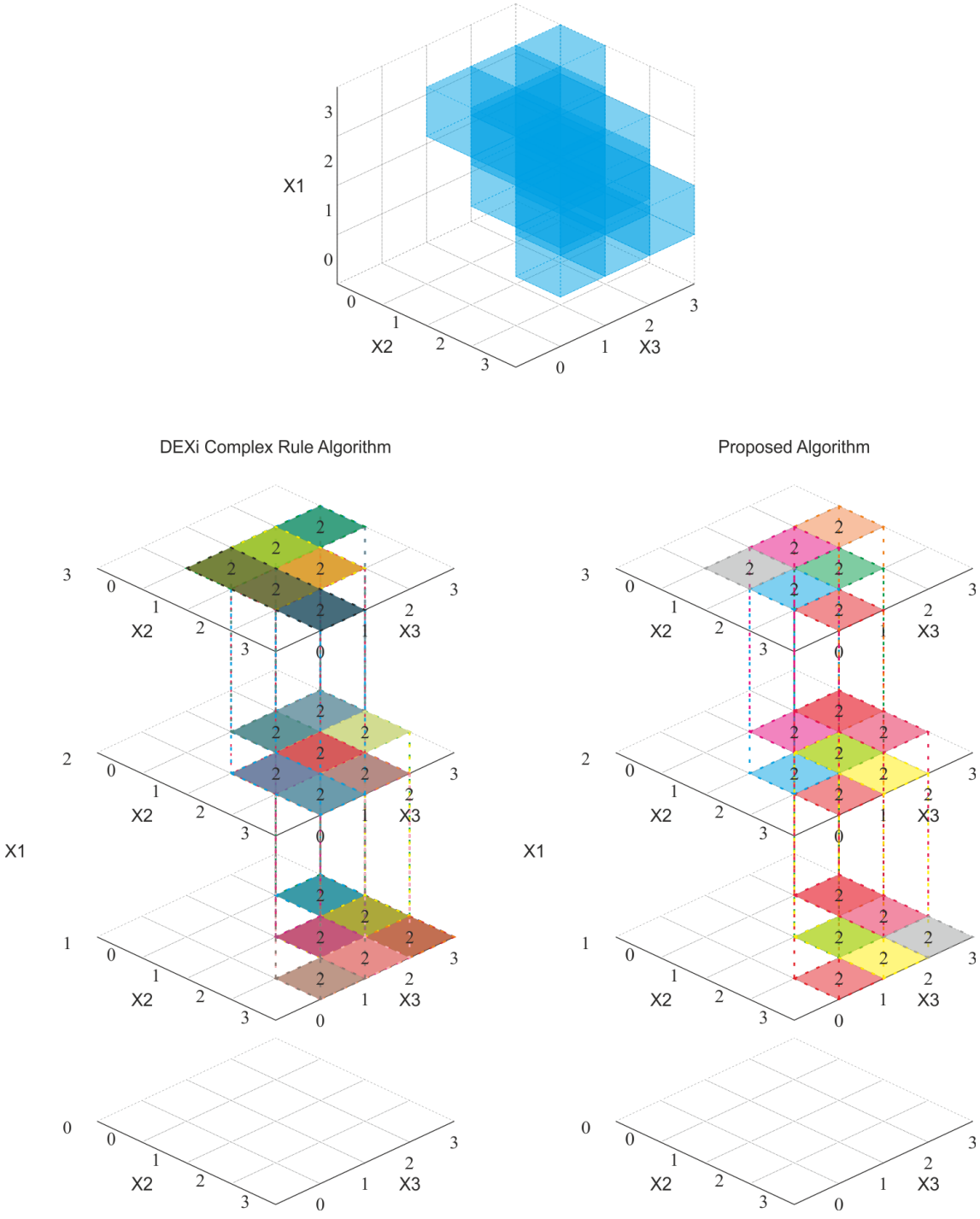


Figure 13: Elementary rules with decision value $O_{4x4x4logic} = 2$ and complex rules with different colors generated from the DEXi complex rule algorithm and from the proposed algorithm

In order to have a better representation of the Fig. 13, the generated complex rules from two algorithms are shown in table by ordinal values, Table 20.

#	X1	X2	X3
1	1	>=1	3
2	1:2	1:2	3
3	>=1	1	3
4	1	>=2	>=2
5	1:2	2	>=2
6	1:2	>=2	2
7	>=1	2	2
8	1	3	>=1
9	1:2	3	1:2
10	>=1	3	1
11	2	1:2	>=2
12	2	>=1	2
13	>=2	1	>=2
14	>=2	1:2	2
15	2	2	>=1
16	2	>=2	1:2
17	>=2	2	1:2
18	>=2	>=2	1
19	3	1	>=1
20	3	1:2	1:2
21	3	>=1	1

(a) Complex rules generated by the DEXi complex rule algorithm

#	X1	X2	X3
1	>=1	3	1
2	>=1	2	2
3	>=2	2	1
4	>=1	1	3
5	>=2	1	2
6	3	1	1
7	1:2	>=2	2
8	1:2	1:2	3
9	1	>=1	3

(b) Complex rules generated by the proposed algorithm

Table 20: Complex rules generated from the two algorithms

The last comparison is based on the running time between two algorithms for the same DEXi models, the DEXi complex rules algorithm in the table is denoted as *DEXi CR* and the proposed algorithm is denoted as *New CR*, Table 21.

#	Running Time		Number of complex rules	
	DEXi CR	New CR	DEXi CR	New CR
1	1.94 sec	0.98 sec	11	26
2	1280 sec	0.395 sec	121	64
3	0.42 sec	0.13 sec	30	18

Table 21: Difference between two algorithm based on running time and number of generated complex rules

6 Conclusion

The DEX method is implemented on the DEXi computer program and it is a decision-making method which is based on qualitative values and it presents its utility function by using decision rules. In order to present the decision rules in a more compact and comprehensive way, DEXi uses the complex rules.

This work presents the problem of complex rules, DEXi complex rule algorithm, the proposed algorithm for generating complex rules and a comparison between two algorithms based on the time complexity, efficiency of the result that they generate and the running time.

Regarding the time complexity the proposed algorithm is $O(nmop)$ which is much more efficient compared to the DEXi complex rules algorithm which is $O(m^n)$. Also, regarding the number of complex rules that the proposed algorithm generates, in most cases it performs better than the actual one. Since the complexity of the proposed algorithm is lower it performs much faster regarding the running time.

The main contribution of this work is the proposed algorithm which is more efficient based on the time complexity and the number of complex rules that it generates together with running time. As part of this work, both algorithms were implemented in the open source library JDEXi V4.

References

Bohanec, M. (2008). Dexi: Program for multi-attribute decision making user's manual. *Ljubljana, Slovenia: Institut Jozef Stefan*.

Bohanec, M. (2017). Multi-criteria dex models: An overview and analysis.

Bohanec, M., Žnidaršič, M., Rajkovič, V., Bratko, I., & Zupan, B. (2013). Dex methodology: three decades of qualitative multi-attribute modeling. *Informatica*, *37*(1).

Bouyssou, D., Marchant, T., Pirlot, M., Tsoukiàs, A., & Vincke, P. (2006). *Evaluation and decision models with multiple criteria: Stepping stones for the analyst* (Vol. 86). Springer Science & Business Media.

Greco, S., Figueira, J., & Ehrgott, M. (2005). Multiple criteria decision analysis. *Springer's International series*.

Jdexi: Open-source dexi java library. (2018). http://kt.ijs.si/MarkoBohanec/jdexi.html. (Accessed: 03/12/2018)

Trdin, N., & Bohanec, M. (2018, Mar 01). Extending the multi-criteria decision making method dex with numeric attributes, value distributions and relational models. *Central European Journal of Operations Research*, *26*(1), 1–41.